Course: HDSDA

Name: Brian Gibbons

Student ID: 13109961

E-mail: brian.gibbons@student.ncirl.ie

Extract and Analyse Player Performance Data from the Fantasy Premier League Website

Table of Contents

Executive Summary

The purpose of this project was to provide a system which could be used as a tool to aid decision making in the Fantasy Premier League football game. This objective was achieved by scraping the Fantasy Premier League (FPL) website of all of the player scoring data and creating an inventive, unique, intuitive and user-friendly dashboard which allowed for easy access to this data. The dashboard also allows users to drill down into the player data to show patterns and trends which are otherwise unavailable from the website.

The overall system comprises of cutting edge technology in the form of the Python programming language, MySQL Relational Database Management System (RDBMS) and Qlikview Business Intelligence software to produce a fully integrated, fully tested and robust platform to allow users the ability to derive actionable insight from the data.

1. Introduction

The Fantasy Premier League game is one of the most widely played fantasy sports games in the world. There are over 3 million players from almost every country in the world taking part in the 2013/14 season (Premier League, 2014). The vast majority of these players come from Ireland and the UK where the majority of the fan base for the Barclays Premier League are situated. However, in recent years the base of players has spread significantly across the USA and South East Asia in particular where the appetite for Premier League football has increased in proportion to the availability of the live matches on television in these countries through large broadcast providers such as Sky and ESPN.

1.1 Purpose

The overall purpose of this project was to create a single integrated platform which would allow users to view useful information and trends on all players in the game which can help managers make informed calls on who the most suitable players are to choose based on past performances. The final platform was an interactive dashboard in Qlikview where the end user can make selections based on teams/positions/players/price etc.

1.2 System Description

There were a number of processes involved in the overall system which are described by the graphic below.

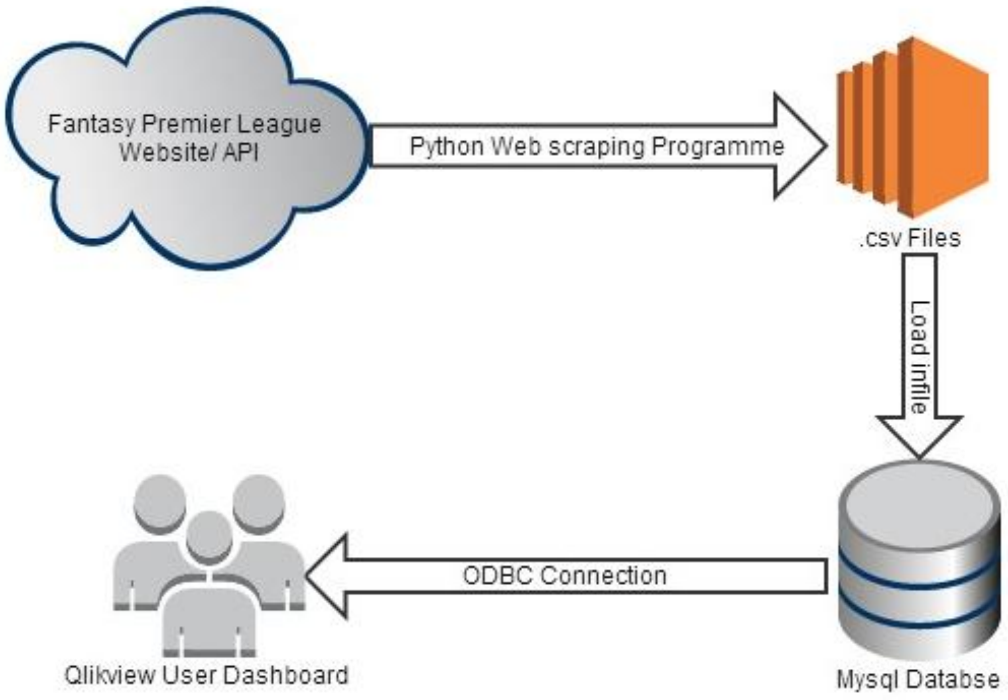


Fig 1.1: Overall System description

The first step in the system was to scrape the website using scripts written in the python programming language. A number of different packages were used to aid in this process. The outputs of these scripts were a number of CSV files with information on all players in the FPL contained within them.

These CSV files were then loaded into a mysql database as structured tables which could be queried using the SQL querying language.

An ODBC connection was then made using an ODBC driver between the mysql database and the Qlikview interface. This allowed us to import all of the data into our qlikview database which in turn enabled us to perform all of the analysis we required. All graphs and tables in qlikview are created from the source data retrieved from the website on that particular date.

1.3 Rules

The premise of the game is simple. You are the manager of your own squad of players. You must choose 15 players from a combination of any of the clubs in the Premier League. There are a number of rules which are applied to ensure a fair and equitable set up for all managers.

1. Every player has an initial value at the start of the season. The initial squad of players chosen originally must have a value of no more than £1m. As the season goes on the price of these players fluctuates based on their performances. The more points a player scores, the more people transferring that player in, the more his price will rise and vice versa.
2. The squad of players consist of 2 goalkeepers, 5 defenders, 5 midfielders and 3 forwards. A minimum of 1 goalkeeper, 3 defenders, 2 midfielders and 1 forward must be played every game week. Based on these rules, a team of 11 players must be chosen to start each game week with 4 substitutes in reserve should any player not play.
3. A maximum of 3 players from any single team can be chosen. This prevents players from overloading their teams with players from the top performing team and forces the manager, along with the price constraints, to purchase cheaper players from teams in lower end of the league.
4. A manager is allowed to make one free transfer every game week. If a manager makes more than one change per game week then 4 points will be deducted from their overall score. If a manager does not make any transfers in a game week then their one free transfer is carried over to the next game week where they will have two free transfers. This does not apply continuously so if the manager does not make a transfer in two consecutive game weeks then they will still only have 2 free transfers available to them
5. During the course of the season a manager will be granted two wildcards. When a manager plays their wildcard it means they can make an unlimited amount of transfers that game week without incurring any point reductions. This effectively enables the manager to change their entire team if they wish to do so. One of the wildcards can be played at any stage during the season while the other can only be played during the busy month of December.

(Premier League, 2014)

1.4 Scoring System

The scoring system is quite simple and it is based around points awarded to players based on the number of goals, assists, clean sheets, bonus points and saves they make, depending on the position they play. The full scoring system taken from the website is shown below.

Action	Points
For playing up to 60 minutes	1
For playing 60 minutes or more	2
For each goal scored by a goalkeeper or defender	6
For each goal scored by a midfielder	5
For each goal scored by a forward	4
For each goal assist	3
For a clean sheet by a goalkeeper or defender	4
For a clean sheet by a midfielder	1
For every 3 shot saves by a goalkeeper	1
For each penalty save	5
For each penalty miss	-2
Bonus points for the best players in a match	1-3
For every 2 goals conceded by a goalkeeper or defender	-1
For each yellow card	-1
For each red card	-3
For each own goal	-2

Fig 1.2: Points Scoring System (Premier League, 2014)

One player can be chosen in the team as captain. The captain will have any points he accrues doubled. It is always a good idea for a manager to choose a player who they think will accumulate the most points as captain. One vice-captain can also be chosen. In the case where the captain does not play, the captaincy will be transferred to the vice-captain.

1.5 Leagues and Prizes

Once a team is entered it is automatically included in the worldwide league against the 3 million plus other managers around the world. The prizes on offer for the overall league are:

"Subject to any unforeseen changes, the Winner's Prize will be a 7-night break in the United Kingdom for two people to include VIP hospitality at two Barclays Premier League matches (the identity of which is determined by The Premier League) and visits to a selection of popular tourist attractions in the UK. The Winner's Prize includes travel and seven nights' hotel accommodation on a bed & breakfast basis"

"The Quarterly Prizes (one each quarter during the season) shall each consist of a VIP Trip for two to a Barclays Premier League match (the identity of which is determined by The Premier League). The prize includes travel, two nights' accommodation including breakfast and two match tickets"

"The Monthly Prizes (one each month during the season) shall each consist of a tablet computer, a Nike Incyte match ball and EA SPORTS FIFA 14 game. The precise specification and nature of these shall be at the discretion of The Premier League" (Premier League, 2014)

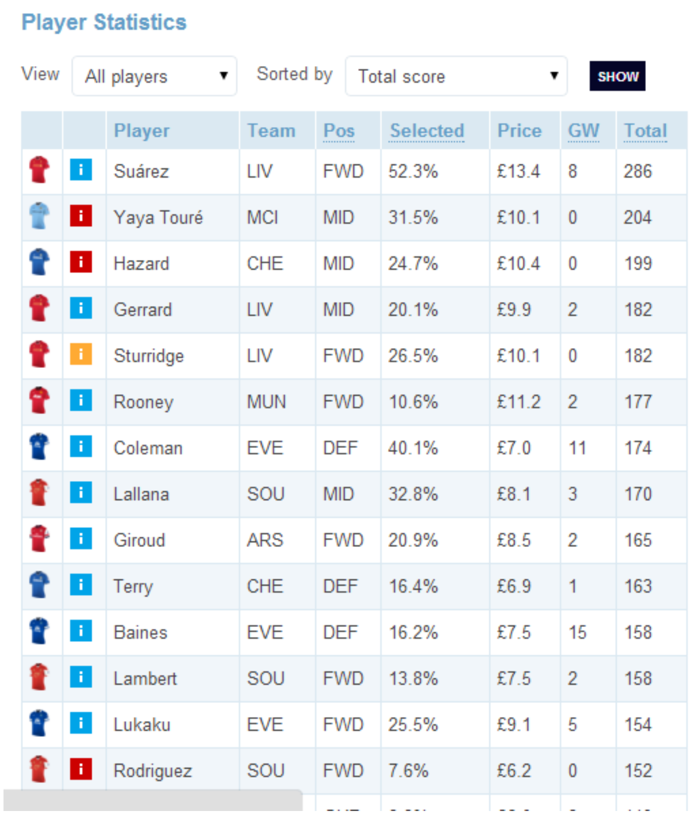
Managers can enter into public leagues which are available to every manager or can enter their own private leagues with friends. Many users of the website create money leagues where every manager in the league must submit a pre-determined sum of money into the overall pot prior to the first game week. The winner of the league is then given the final accumulated sum at the end of the competition. These money leagues are entirely governed by the members of the league and no authorisation needs to be given by the Premier League for these to take place.

2. Literature Review

The main motivation behind the creation of this system was a lack of rival systems which could give the user the information they required to assist them in the tasks involved with managing a fantasy football team. These tasks include but are not limited to transfers, substitutions, captaincy choices, formation selection etc.

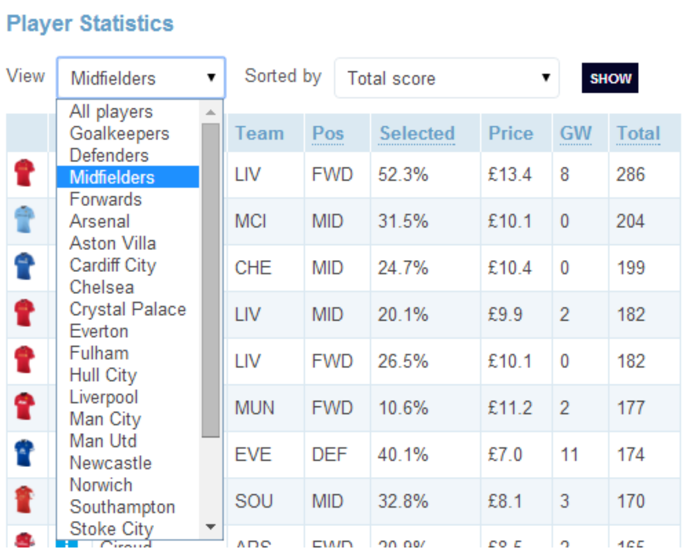
2.1 Fantasy Premier League Website

The fantasy premier league website does provide a screen which provides users with information which helps managers make decisions on these aspects of the game. However, the look and feel of the screen is not very intuitive and all data is represented as numbers and text. Due to the lack of graphical data it can be difficult for a manager to spot trends and analyse a player's performance adequately as the human eye responds better to graphical information than numeric or textual information. In fact 90% of information transmitted to the brain is visual, and visuals are processed 60,000X faster in the brain than text (J6 Design, 2012). An example of the data available on the website is shown below.



		Player	Team	Pos	Selected	Price	GW	Total
		Suárez	LIV	FWD	52.3%	£13.4	8	286
		Yaya Touré	MCI	MID	31.5%	£10.1	0	204
		Hazard	CHE	MID	24.7%	£10.4	0	199
		Gerrard	LIV	MID	20.1%	£9.9	2	182
		Sturridge	LIV	FWD	26.5%	£10.1	0	182
		Rooney	MUN	FWD	10.6%	£11.2	2	177
		Coleman	EVE	DEF	40.1%	£7.0	11	174
		Lallana	SOU	MID	32.8%	£8.1	3	170
		Giroud	ARS	FWD	20.9%	£8.5	2	165
		Terry	CHE	DEF	16.4%	£6.9	1	163
		Baines	EVE	DEF	16.2%	£7.5	15	158
		Lambert	SOU	FWD	13.8%	£7.5	2	158
		Lukaku	EVE	FWD	25.5%	£9.1	5	154
		Rodriguez	SOU	FWD	7.6%	£6.2	0	152

Fig 2.1: FPL Player statistics (Premier League, 2014)



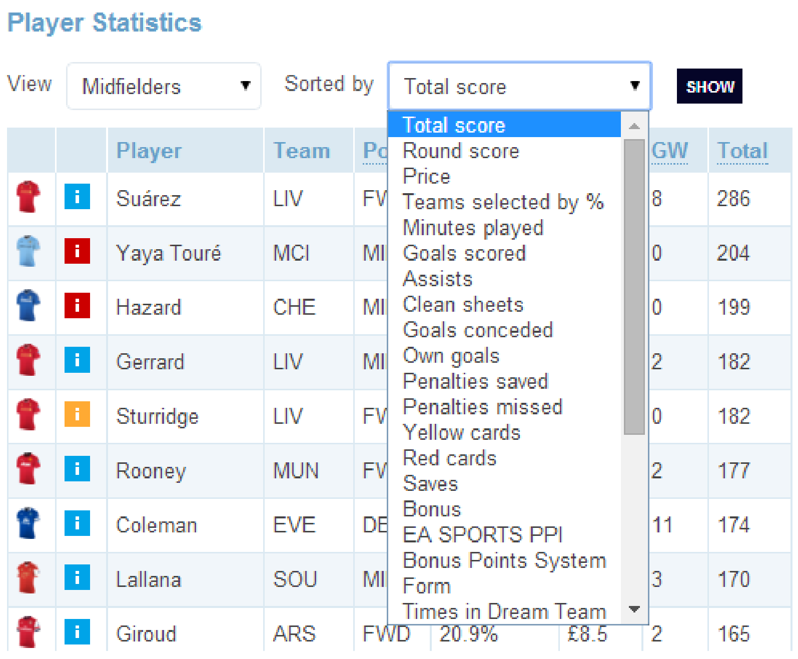
Player Statistics

View: Midfielders ▼ Sorted by: Total score ▼ SHOW

Team	Pos	Selected	Price	GW	Total
LIV	FWD	52.3%	£13.4	8	286
MCI	MID	31.5%	£10.1	0	204
CHE	MID	24.7%	£10.4	0	199
LIV	MID	20.1%	£9.9	2	182
LIV	FWD	26.5%	£10.1	0	182
MUN	FWD	10.6%	£11.2	2	177
EVE	DEF	40.1%	£7.0	11	174
SOU	MID	32.8%	£8.1	3	170

Fig 2.2: Statistical information from the Fantasy Football website (Premier League, 2014)

As we can see, the player's information is shown in a tabular format and players can be chosen based on position or team name. A range of Key Performance Indicators (KPIs) are also available to the user in the "Sorted by" dropdown box in the top right hand corner.



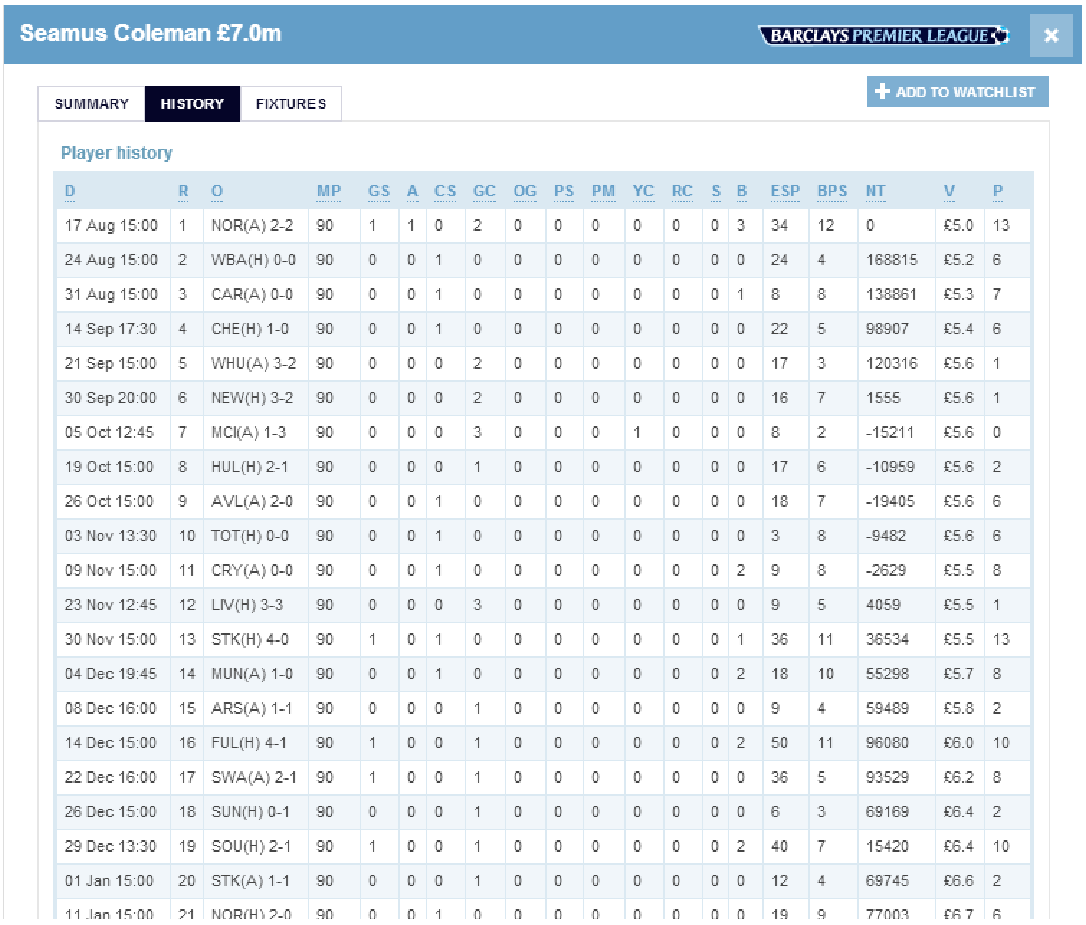
Player Statistics

View: Midfielders ▼ Sorted by: Total score ▼ SHOW

Player	Team	Po	GW	Total
Suárez	LIV	FV	8	286
Yaya Touré	MCI	MI	0	204
Hazard	CHE	MI	0	199
Gerrard	LIV	MI	2	182
Sturridge	LIV	FV	0	182
Rooney	MUN	FV	2	177
Coleman	EVE	DE	11	174
Lallana	SOU	MI	3	170
Giroud	ARS	FWD	2	165

Fig 2.3: KPIs on Fantasy Premier League screen (Premier League, 2014)

These KPIs all contain very useful information. However, it becomes difficult for a user to compare players with each other and a time series of scoring history is only available when an individual player is selected which makes it difficult to identify patterns in the data as can be seen below.


Seamus Coleman £7.0m

Player history

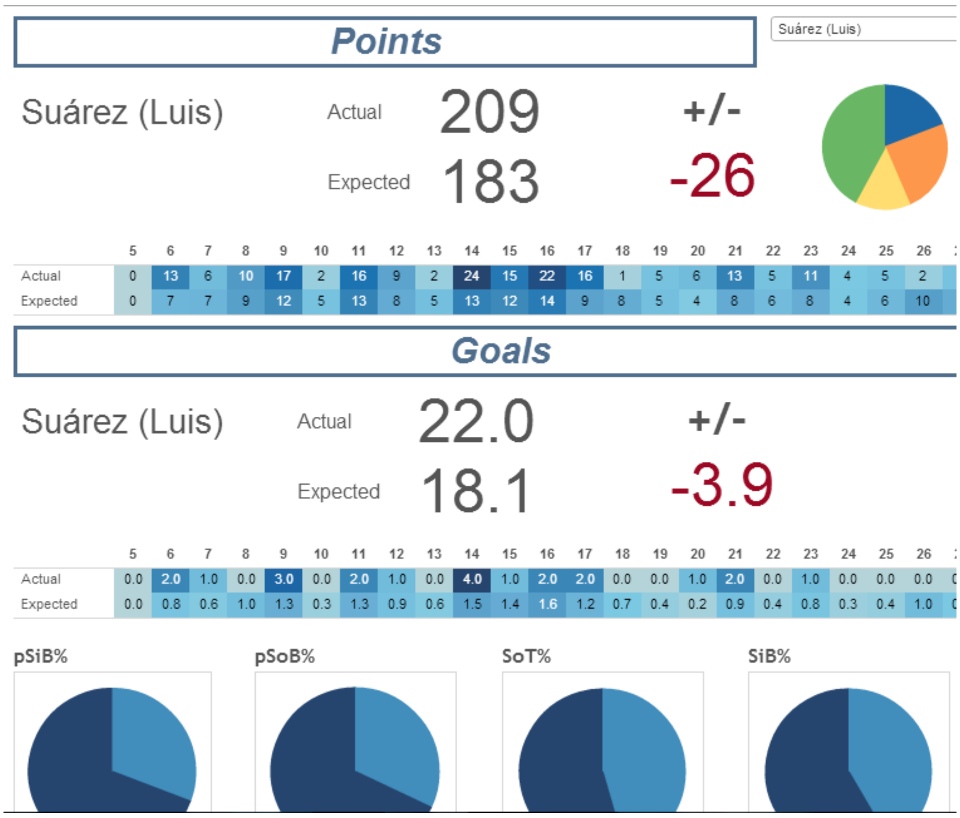
D	R	O	MP	GS	A	CS	GC	OG	PS	PM	YC	RC	S	B	ESP	BPS	NT	V	P
17 Aug 15:00	1	NOR(A) 2-2	90	1	1	0	2	0	0	0	0	0	0	3	34	12	0	£5.0	13
24 Aug 15:00	2	WBA(H) 0-0	90	0	0	1	0	0	0	0	0	0	0	0	24	4	168815	£5.2	6
31 Aug 15:00	3	CAR(A) 0-0	90	0	0	1	0	0	0	0	0	0	0	1	8	8	138861	£5.3	7
14 Sep 17:30	4	CHE(H) 1-0	90	0	0	1	0	0	0	0	0	0	0	0	22	5	98907	£5.4	6
21 Sep 15:00	5	WHU(A) 3-2	90	0	0	0	2	0	0	0	0	0	0	0	17	3	120316	£5.6	1
30 Sep 20:00	6	NEW(H) 3-2	90	0	0	0	2	0	0	0	0	0	0	0	16	7	1555	£5.6	1
05 Oct 12:45	7	MCI(A) 1-3	90	0	0	0	3	0	0	0	1	0	0	0	8	2	-15211	£5.6	0
19 Oct 15:00	8	HUL(H) 2-1	90	0	0	0	1	0	0	0	0	0	0	0	17	6	-10959	£5.6	2
26 Oct 15:00	9	AVL(A) 2-0	90	0	0	1	0	0	0	0	0	0	0	0	18	7	-19405	£5.6	6
03 Nov 13:30	10	TOT(H) 0-0	90	0	0	1	0	0	0	0	0	0	0	0	3	8	-9482	£5.6	6
09 Nov 15:00	11	CRY(A) 0-0	90	0	0	1	0	0	0	0	0	0	0	2	9	8	-2629	£5.5	8
23 Nov 12:45	12	LIV(H) 3-3	90	0	0	0	3	0	0	0	0	0	0	0	9	5	4059	£5.5	1
30 Nov 15:00	13	STK(H) 4-0	90	1	0	1	0	0	0	0	0	0	0	1	36	11	36534	£5.5	13
04 Dec 19:45	14	MUN(A) 1-0	90	0	0	1	0	0	0	0	0	0	0	2	18	10	55298	£5.7	8
08 Dec 16:00	15	ARS(A) 1-1	90	0	0	0	1	0	0	0	0	0	0	0	9	4	59489	£5.8	2
14 Dec 15:00	16	FUL(H) 4-1	90	1	0	0	1	0	0	0	0	0	0	2	50	11	96080	£6.0	10
22 Dec 16:00	17	SWA(A) 2-1	90	1	0	0	1	0	0	0	0	0	0	0	36	5	93529	£6.2	8
26 Dec 15:00	18	SUN(H) 0-1	90	0	0	0	1	0	0	0	0	0	0	0	6	3	69169	£6.4	2
29 Dec 13:30	19	SOU(H) 2-1	90	1	0	0	1	0	0	0	0	0	0	2	40	7	15420	£6.4	10
01 Jan 15:00	20	STK(A) 1-1	90	0	0	0	1	0	0	0	0	0	0	0	12	4	69745	£6.6	2
11 Jan 15:00	21	NOR(H) 2-0	90	0	0	1	0	0	0	0	0	0	0	0	19	9	77003	£6.7	6

Fig 2.4: Player History Data (Premier League, 2014)

This is not an ideal view of the data as it is very hard to spot trends and does not give the manager much of an indication of the potential for future points scoring without doing further manual analysis themselves.

2.2 PL Fantasy Blog

A similar type project has already been completed which is called PLFantasy. This is a blog created by an unknown user who has created an information dashboard using the tableau software (PLFantasy, 2014). The dashboard is viewed from a player level and the user can select whatever player they would like for analysis purposes. The dashboard can be downloaded from the main website and viewed in tableau public format (Tableau, 2014).



Points

Suárez (Luis)

Actual 209

Expected 183

+/- -26

	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Actual	0	13	6	10	17	2	16	9	2	24	15	22	16	1	5	6	13	5	11	4	5	2
Expected	0	7	7	9	12	5	13	8	5	13	12	14	9	8	5	4	8	6	8	4	6	10

Goals

Suárez (Luis)

Actual 22.0

Expected 18.1

+/- -3.9

	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Actual	0.0	2.0	1.0	0.0	3.0	0.0	2.0	1.0	0.0	4.0	1.0	2.0	2.0	0.0	0.0	1.0	2.0	0.0	1.0	0.0	0.0	0.0
Expected	0.0	0.8	0.6	1.0	1.3	0.3	1.3	0.9	0.6	1.5	1.4	1.6	1.2	0.7	0.4	0.2	0.9	0.4	0.8	0.3	0.4	1.0

Fig 2.5: PLFantasy Dashboard (PLFantasy, 2014)

This dashboard provides a more graphical interface to the user and many of the KPIs stand out better than they do in the Premier League website screen. However, a lot of the information in the dashboard would not be very relevant to FPL managers and a lot of the data shown is not very well explained.

If a user wishes to have full access to the dashboard they must sign up to the blog. This would turn a lot of users away as they would only be interested in the information contained within the dashboard itself.

For these reasons, a gap in the market was identified for a program where users of the game could have instant access to all of the information they need to make the correct team selections without having to sign up or pay for any extra add-ons.

3. Data Extraction from FPL Website

All of the data which we needed for analysis was contained within the FPL website. Any user can go in and try and manually extract all of this data if they so wish. However, with over 600 players in the game database, this would be a very time intensive activity and would have to be repeated after every game week had completed. It was decided that a computer programme would be written to carry out this task automatically which would be run at the end of every game week, extracting all of the information required from the website. The python programming language was used for this task.

3.1 Python

The python programming language is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries.

It is a high-level general-purpose programming language that can be applied to many different classes of problems. The language comes with a large standard library that covers areas such as string processing (regular expressions, Unicode, calculating differences between files), Internet protocols (HTTP, FTP, SMTP, XML-RPC, POP, IMAP, CGI programming), software engineering (unit testing, logging, profiling, parsing Python code), and operating system interfaces (system calls, file systems, TCP/IP sockets). Python is an open source programming language and can be downloaded completely free of charge onto any machine (Python, 2014).

3.1.1 Why Python?

There were a number of reasons python was chosen as the language to carry out all of the data scraping work.

1. It is free. This enabled us to download the packages online and install them on our machines immediately. Work could begin as soon as the set up was complete
2. It is easy to learn. Python's simple and straight-forward syntax also encourages good programming habits, especially through its focus on white space indentation, which contributes to the development of neat looking code (Six Feet Up, 2013)
3. Python is very good at handling and parsing strings. It contains a number of very useful libraries which help the programmer scrape data from websites. Almost all of the data to be scraped for this project was in the form of strings.

3.1.2 Version

There are currently two up to date versions of python available for installation. Version 3.x was created in 2008 and is still under active development and has already seen over five years of stable releases, including version 3.3 in 2012 and 3.4 in 2014. The final 2.x, version 2.7 release came out in mid-2010, with a statement of extended support for this end-of-life release. The 2.x branch will see no new major releases after that (Wiki Python, 2014).

There are a couple of reasons why programmers would choose version 2.x over the more up to date 3.x version.

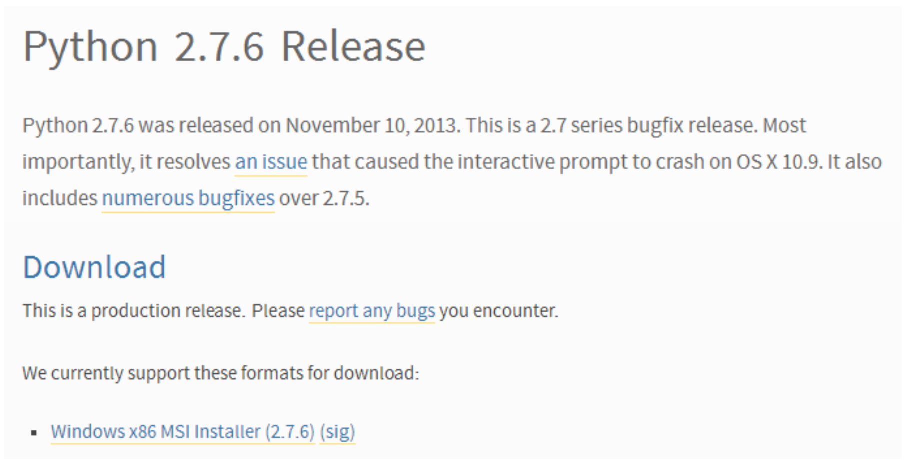
1. If you're deploying to an environment you don't control, that may impose a specific version, rather than allowing you a free selection from the available versions
2. If you want to use a specific third party package or utility that doesn't yet have a released version that is compatible with Python 3, and porting that package is a non-trivial task, you may choose to use Python 2 in order to retain access to that package (Wiki Python, 2014).

For the purposes of this project it was decided to use the 2.7.6 version of python. One of the main reasons for this was that the version on computers in the college is 2.7.6. This meant that work could be completed in the college as well as at home.

3.1.3 Installation

As the machine which was used for most of the project tasks used the Windows operating system, an installation procedure for python specific for Windows was used. Windows does not require Python natively and thus does not pre-install a version of Python.

The package can be downloaded from the python.org website. For our case the 2.7.6 release was downloaded.



Python 2.7.6 Release

Python 2.7.6 was released on November 10, 2013. This is a 2.7 series bugfix release. Most importantly, it resolves an issue that caused the interactive prompt to crash on OS X 10.9. It also includes numerous bugfixes over 2.7.5.

Download

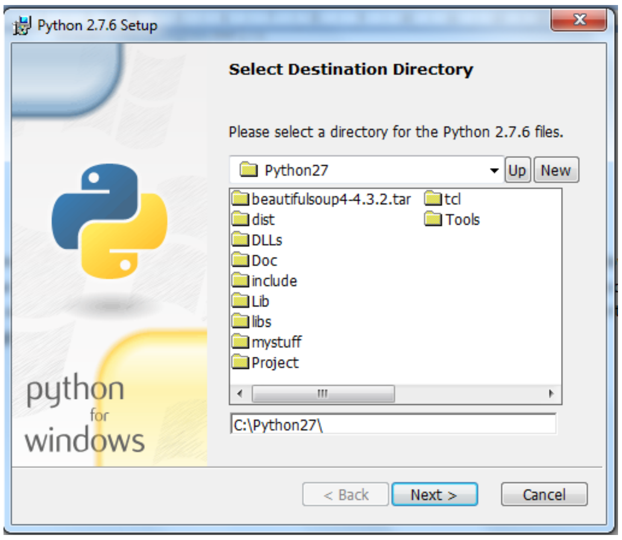
This is a production release. Please report any bugs you encounter.

We currently support these formats for download:

- Windows x86 MSI Installer (2.7.6) (sig)

Fig 3.1: Python download (Python, 2014)

The Windows x86 MSI Installer (2.7.6) (sig) file was downloaded. On completion of the download, the .exe file is run and the folder is saved into the computer C drive by default.



Select Destination Directory

Please select a directory for the Python 2.7.6 files.

Python27

beautifulsoup4-4.3.2.tar, dist, DLLs, Doc, include, Lib, libs, mystuff, Project, tcl, Tools

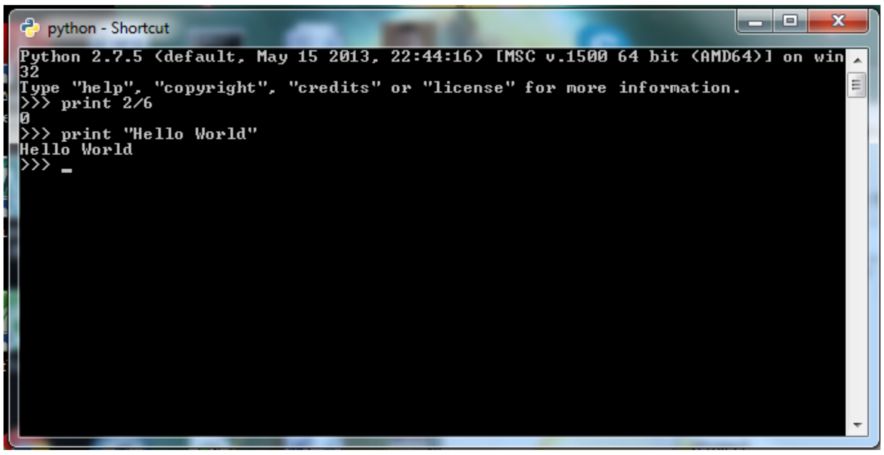
C:\Python27\

Fig 3.2: Python download

This can be changed by the user if they so wish. Once this has been completed, the user can open the python terminal and begin writing code.

3.1.4 Executing Code in Python

There are a number of different ways to execute code within python. The first and least used method is to write the code into the python terminal. When a statement is written and executed in the terminal the results of the statement are returned within the same terminal.

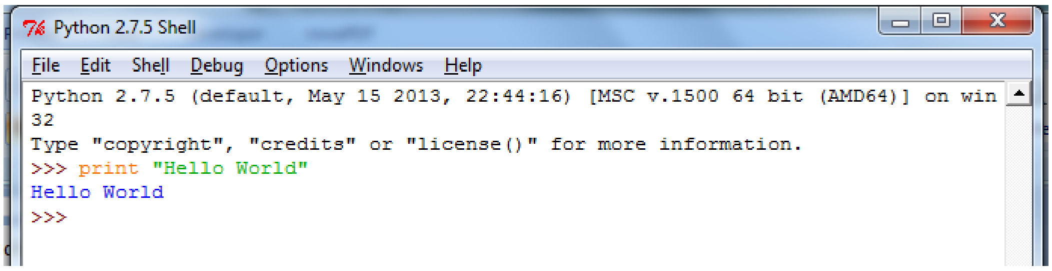


```
Python 2.7.5 (default, May 15 2013, 22:44:16) [MSC v.1500 64 bit (AMD64)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> print 2/6
0
>>> print "Hello World"
Hello World
>>> _
```

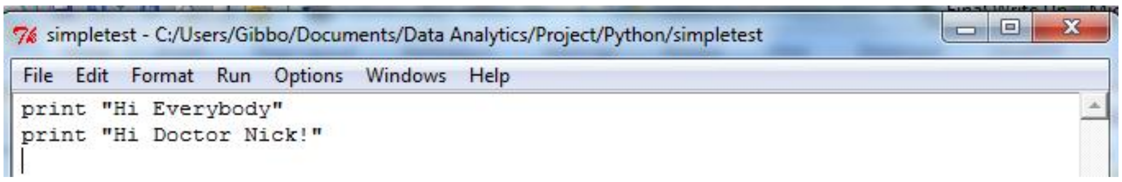
Fig 3.3: Python terminal

This type of code writing can be useful for testing certain lines of code to give us answers straight away without having to run full scripts. However, when attempting to write sequential code in script format to run iteratively through a sequence of statements, other alternatives should be used.

One such method is to use the IDLE GUI for python which enables the user to run code straight from the terminal or to create scripts which can be run sequentially.

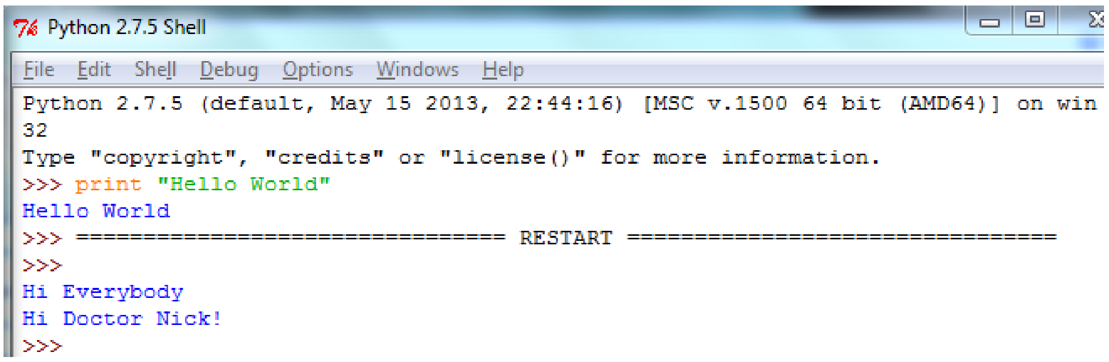


```
Python 2.7.5 (default, May 15 2013, 22:44:16) [MSC v.1500 64 bit (AMD64)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> print "Hello World"
Hello World
>>>
```



simpletest - C:/Users/Gibbo/Documents/Data Analytics/Project/Python/simpletest
File Edit Format Run Options Windows Help

```
print "Hi Everybody"
print "Hi Doctor Nick!"
```



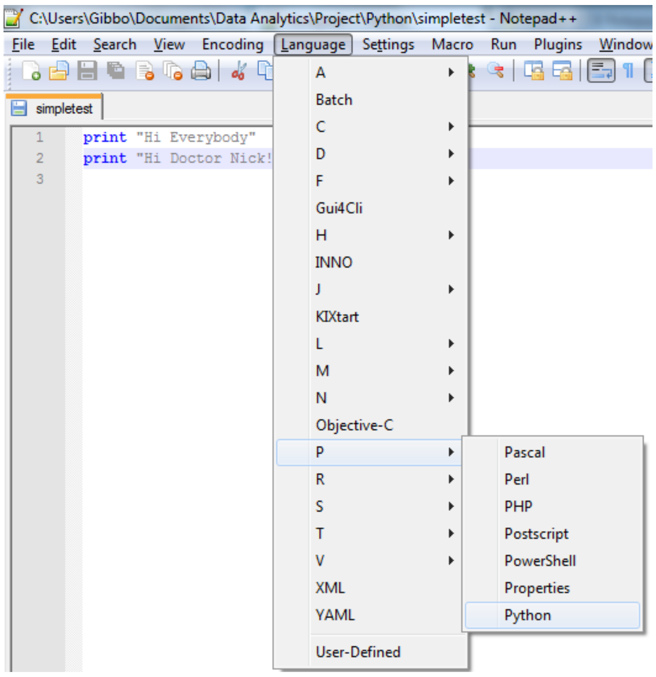
```
Python 2.7.5 (default, May 15 2013, 22:44:16) [MSC v.1500 64 bit (AMD64)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> print "Hello World"
Hello World
>>> ================================ RESTART ================================
>>>
Hi Everybody
Hi Doctor Nick!
>>>
```

Fig 3.4: Python IDLE editor and terminal

In the example above, we create a script to print out to sentences to the terminal. In order to run the script F5 must be pressed and the output will be shown on the IDLE GUI.

The method used in this project for writing scripts was to run code from a python text file through the windows command prompt. This involved writing the script in Notepad++, saving it as a python file and running that file through the command prompt.

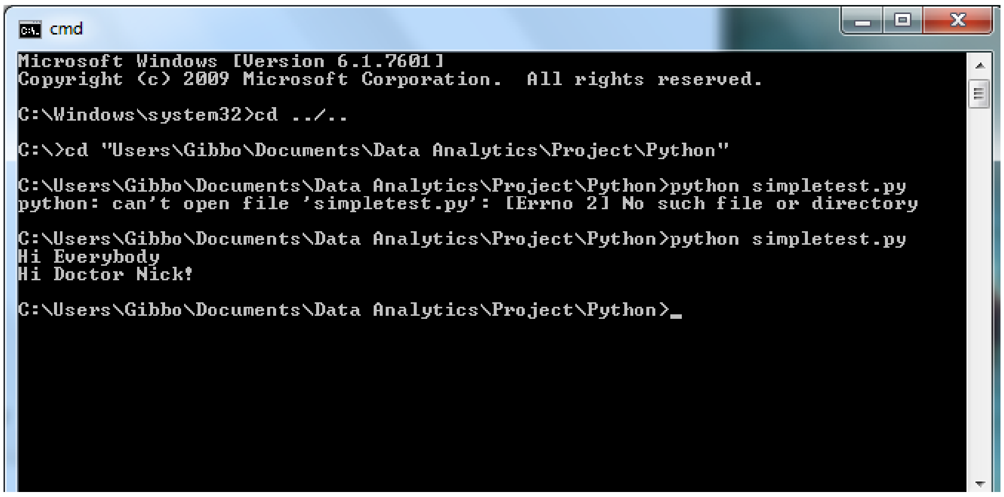
All of the code written for this program was done in Notepad++. Notepad++ is a free source code editor and Notepad replacement that supports several languages (Notepad ++, 2014). When writing a script in python, you can select the python language from the drop down and this highlights all of the key words making the script much more legible.


C:\Users\Gibbo\Documents\Data Analytics\Project\Python\simpletest - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window
simpletest
A
Batch
C
D
F
Gui4Cli
H
INNO
J
KIXtart
L
M
N
Objective-C
P
R
S
T
V
XML
YAML
User-Defined
Pascal
Perl
PHP
Postscript
PowerShell
Properties
Python

```
print "Hi Everybody"
print "Hi Doctor Nick!
```

Fig 3.5: Notepad++ Interface

In order to run this script we must then open up the windows command prompt and cd to the directory where we have saved the file.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>cd ../..

C:\>cd "Users\Gibbo\Documents\Data Analytics\Project\Python"

C:\Users\Gibbo\Documents\Data Analytics\Project\Python>python simpletest.py
python: can't open file 'simpletest.py': [Errno 2] No such file or directory

C:\Users\Gibbo\Documents\Data Analytics\Project\Python>python simpletest.py
Hi Everybody
Hi Doctor Nick!

C:\Users\Gibbo\Documents\Data Analytics\Project\Python>_
```

Fig 3.6: Windows Command Prompt

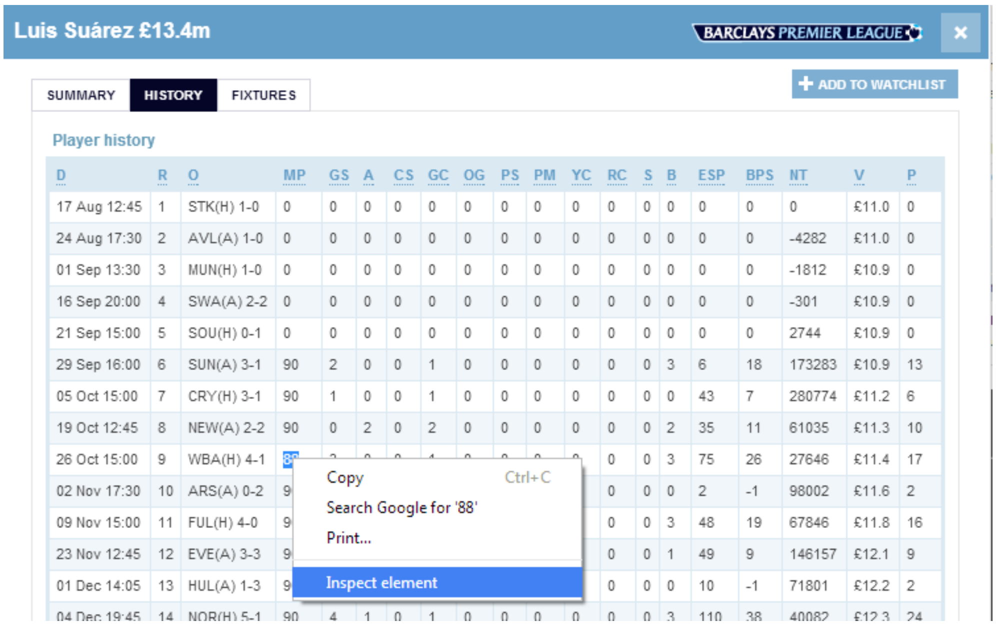
As we can see the when we execute the code using the command - python “scriptname.py” – the results from the script are output to the command prompt. This was the method used to run all of the scripts in the system.

3.2 Scraping the Data

3.2.1 Web Page Layout

All of the data that we require from the FPL website is contained in each individual players scoring history. In order to access this through the website we must go to the transfers screen and select a player for whom we wish to see their previous scores.

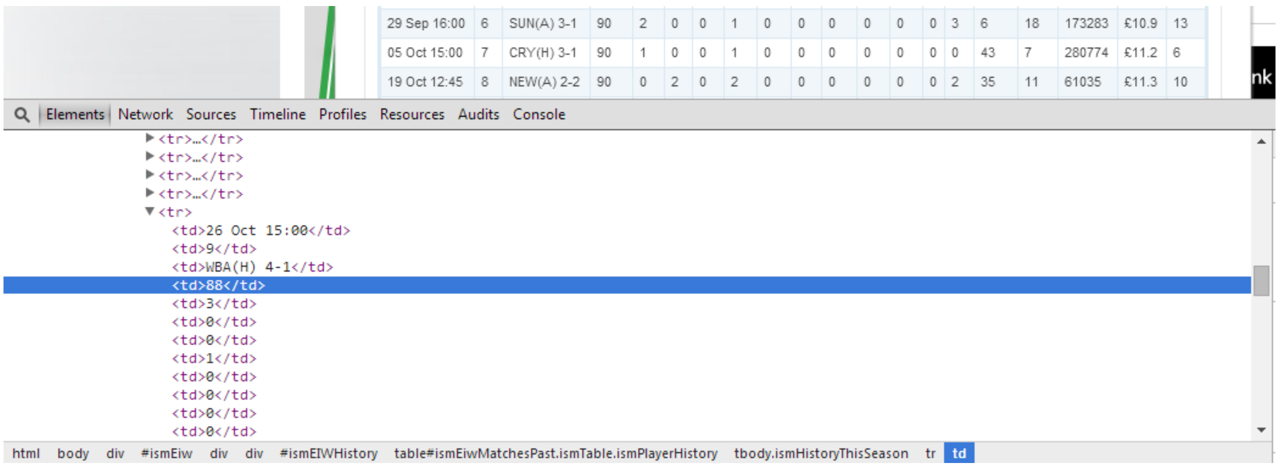
In order to see how this data is sent to the client we can highlight a certain part of the text and select "Inspect Element" as below.



D	R	O	MP	GS	A	CS	GC	OG	PS	PM	YC	RC	S	B	ESP	BPS	NT	V	P
17 Aug 12:45	1	STK(H) 1-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	£11.0	0
24 Aug 17:30	2	AVL(A) 1-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-4282	£11.0	0
01 Sep 13:30	3	MUN(H) 1-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1812	£10.9	0
16 Sep 20:00	4	SWA(A) 2-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-301	£10.9	0
21 Sep 15:00	5	SOU(H) 0-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2744	£10.9	0
29 Sep 16:00	6	SUN(A) 3-1	90	2	0	0	1	0	0	0	0	0	0	3	6	18	173283	£10.9	13
05 Oct 15:00	7	CRY(H) 3-1	90	1	0	0	1	0	0	0	0	0	0	0	43	7	280774	£11.2	6
19 Oct 12:45	8	NEW(A) 2-2	90	0	2	0	2	0	0	0	0	0	0	2	35	11	61035	£11.3	10
26 Oct 15:00	9	WBA(H) 4-1	88	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	3	75	26	27646	£11.4	17
02 Nov 17:30	10	ARS(A) 0-2	9	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	2	-1	98002	£11.6	2
09 Nov 15:00	11	FUL(H) 4-0	9	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	3	48	19	67846	£11.8	16
23 Nov 12:45	12	EVE(A) 3-3	9	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	1	49	9	146157	£12.1	9
01 Dec 14:05	13	HUL(A) 1-3	9	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	10	-1	71801	£12.2	2
04 Dec 19:45	14	NOR(H) 5-1	90	4	1	0	1	0	0	0	0	0	0	3	110	38	40082	£12.3	24

Fig 3.7: Inspect element Google developer

This brings up the HTML source code behind the website and allows the user to view the structure of the web page.



29 Sep 16:00	6	SUN(A) 3-1	90	2	0	0	1	0	0	0	0	0	0	3	6	18	173283	£10.9	13
05 Oct 15:00	7	CRY(H) 3-1	90	1	0	0	1	0	0	0	0	0	0	0	43	7	280774	£11.2	6
19 Oct 12:45	8	NEW(A) 2-2	90	0	2	0	2	0	0	0	0	0	0	2	35	11	61035	£11.3	10

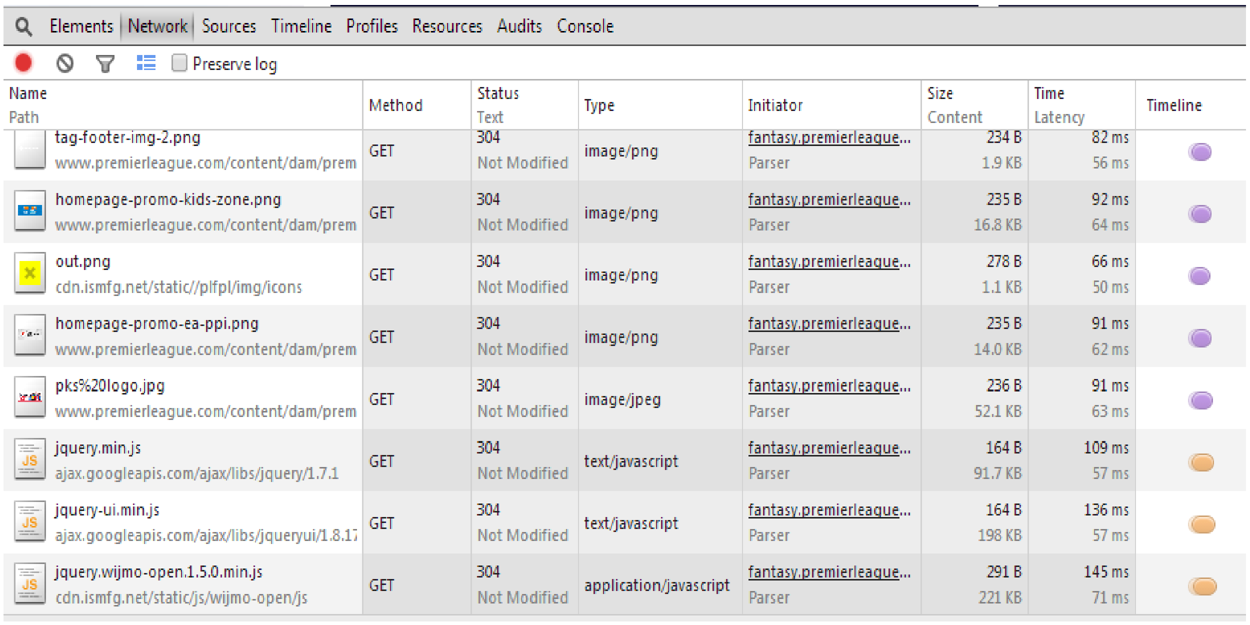
```
▶<tr>…</tr>
▶<tr>…</tr>
▶<tr>…</tr>
▶<tr>…</tr>
▼<tr>
   <td>26 Oct 15:00</td>
   <td>9</td>
   <td>WBA(H) 4-1</td>
   <td>88</td>
   <td>3</td>
   <td>0</td>
   <td>0</td>
   <td>1</td>
   <td>0</td>
   <td>0</td>
   <td>0</td>
   <td>0</td>
```

```
html  body  div  #ismEiw  div  div  #ismEIWHistory  table#ismEiwMatchesPast.ismTable.ismPlayerHistory  tbody.ismHistoryThisSeason  tr  td
```

Fig 3.8: Background HTML code

It is possible to extract all of the information we need using this HTML code. However, this would take a very long time as any program written would have to scrape every item of information for every single player in the game. It would be much more beneficial if we could find the underlying file sent to the webpage which contains all of the data for each individual player.

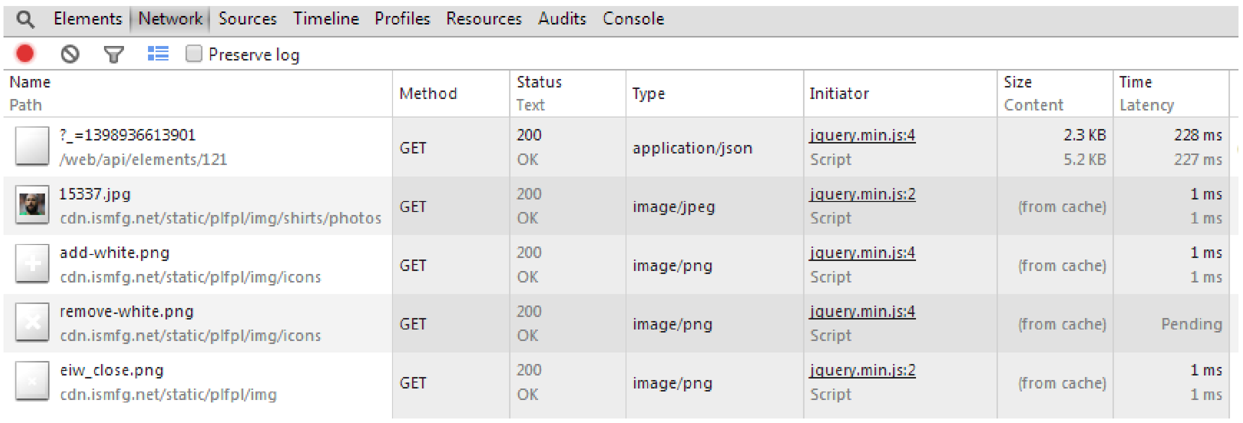
If we click on the Network tab in the developer screen we can see the data which is sent between the server and the client. This contains all of the information which can be viewed on the webpage such as text files, javascript files and images.



Name / Path	Method	Status / Text	Type	Initiator	Size / Content	Time / Latency	Timeline
tag-footer-img-2.png www.premierleague.com/content/dam/prem	GET	304 Not Modified	image/png	fantasy.premierleague... Parser	234 B 1.9 KB	82 ms 56 ms	●
homepage-promo-kids-zone.png www.premierleague.com/content/dam/prem	GET	304 Not Modified	image/png	fantasy.premierleague... Parser	235 B 16.8 KB	92 ms 64 ms	●
out.png cdn.ismfg.net/static//plfpl/img/icons	GET	304 Not Modified	image/png	fantasy.premierleague... Parser	278 B 1.1 KB	66 ms 50 ms	●
homepage-promo-ea-ppi.png www.premierleague.com/content/dam/prem	GET	304 Not Modified	image/png	fantasy.premierleague... Parser	235 B 14.0 KB	91 ms 62 ms	●
pks%20logo.jpg www.premierleague.com/content/dam/prem	GET	304 Not Modified	image/jpeg	fantasy.premierleague... Parser	236 B 52.1 KB	91 ms 63 ms	●
jquery.min.js ajax.googleapis.com/ajax/libs/jquery/1.7.1	GET	304 Not Modified	text/javascript	fantasy.premierleague... Parser	164 B 91.7 KB	109 ms 57 ms	●
jquery-ui.min.js ajax.googleapis.com/ajax/libs/jqueryui/1.8.17	GET	304 Not Modified	text/javascript	fantasy.premierleague... Parser	164 B 198 KB	136 ms 57 ms	●
jquery.wijmo-open.1.5.0.min.js cdn.ismfg.net/static/js/wijmo-open/js	GET	304 Not Modified	application/javascript	fantasy.premierleague... Parser	291 B 221 KB	145 ms 71 ms	●

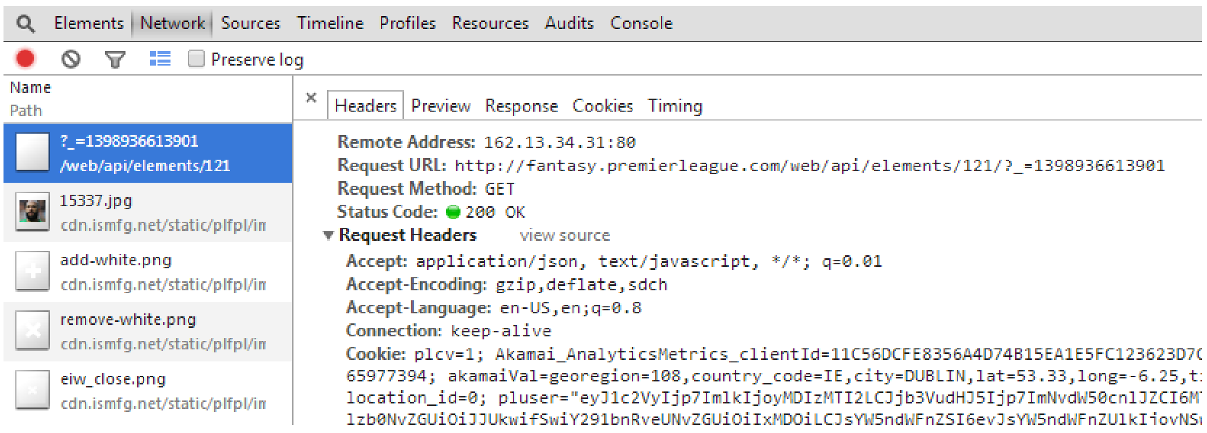
Fig 3.9: Files sent between server and client

If we clear the screen of all the data and click on a player in the Transfers section of the webpage we should be able to limit what is sent between the server and the client to just information about that individual player. This is shown to us in the form of the API for the fantasy premier league website as we can see below.



Name / Path	Method	Status / Text	Type	Initiator	Size / Content	Time / Latency
?_=1398936613901 /web/api/elements/121	GET	200 OK	application/json	jquery.min.js:4 Script	2.3 KB 5.2 KB	228 ms 227 ms
15337.jpg cdn.ismfg.net/static/plfpl/img/shirts/photos	GET	200 OK	image/jpeg	jquery.min.js:2 Script	(from cache)	1 ms 1 ms
add-white.png cdn.ismfg.net/static/plfpl/img/icons	GET	200 OK	image/png	jquery.min.js:4 Script	(from cache)	1 ms 1 ms
remove-white.png cdn.ismfg.net/static/plfpl/img/icons	GET	200 OK	image/png	jquery.min.js:4 Script	(from cache)	Pending
eiw_close.png cdn.ismfg.net/static/plfpl/img	GET	200 OK	image/png	jquery.min.js:2 Script	(from cache)	1 ms 1 ms

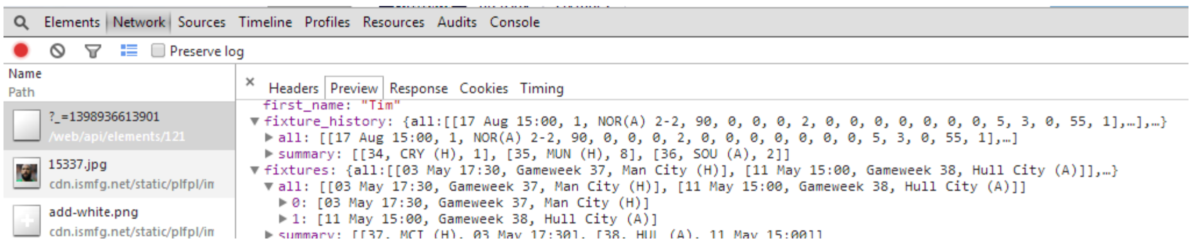
The file that interests us here is the first file using the GET method which is an application/json type file. By clicking on this file we can see all the information relating to it.



```
Remote Address: 162.13.34.31:80
Request URL: http://fantasy.premierleague.com/web/api/elements/121/?_=1398936613901
Request Method: GET
Status Code: 200 OK
▼Request Headers
 Accept: application/json, text/javascript, */*; q=0.01
 Accept-Encoding: gzip,deflate,sdch
 Accept-Language: en-US,en;q=0.8
 Connection: keep-alive
 Cookie: plcv=1; Akamai_AnalyticsMetrics_clientId=11C56DCFE8356A4D74B15EA1E5FC123623D7C
 65977394; akamaiVal=georegion=108,country_code=IE,city=DUBLIN,lat=53.33,long=-6.25,t:
 location_id=0; pluser="eyJ1c2VyIjp7ImlkIjoyMDIzMTI2LCJjb3VudHJ5Ijp7ImNvdW50cnlJZCI6M
 1zb0NvZGUiOiJJUkwifSwiY291bnRveUNvZGUiOiIxMDOiLCJsYW5ndWFnZSI6evJsYW5ndWFnZUlkIiovNS
```

Fig 3.10: FPL API

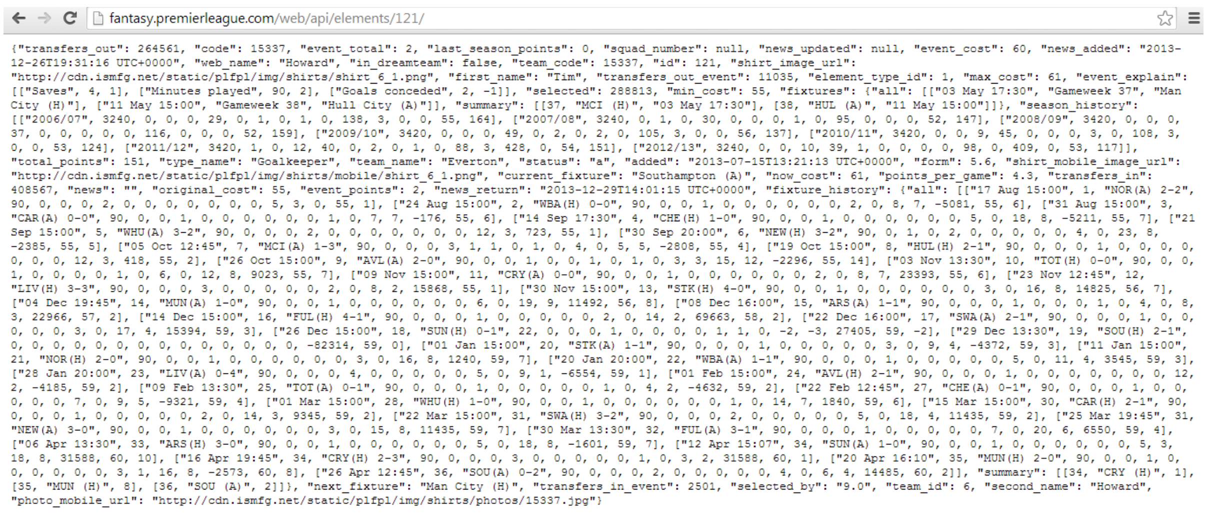
We are interested in finding the data which relates to the players scoring history and future fixture list. If we click on the preview tab we can see that the field relating to these features in the json file are called fixture_history and fixtures.



```
 first_name: "Tim"
▼fixture_history: {all:[[17 Aug 15:00, 1, NOR(A) 2-2, 90, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 5, 3, 0, 55, 1],…],…}
 ▶all: [[17 Aug 15:00, 1, NOR(A) 2-2, 90, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 5, 3, 0, 55, 1],…]
 ▶summary: [[34, CRY (H), 1], [35, MUN (H), 8], [36, SOU (A), 2]]
▼fixtures: {all:[[03 May 17:30, Gameweek 37, Man City (H)], [11 May 15:00, Gameweek 38, Hull City (A)]],…}
 ▼all: [[03 May 17:30, Gameweek 37, Man City (H)], [11 May 15:00, Gameweek 38, Hull City (A)]]
  ▶0: [03 May 17:30, Gameweek 37, Man City (H)]
  ▶1: [11 May 15:00, Gameweek 38, Hull City (A)]
 ▶summary: [[37, MCI (H), 03 May 17:30], [38, HUL (A), 11 May 15:00]]
```

Fig 3.11: API Overview

In order to extract this data from the website for every player we needed to find a unique identifier for the file relating to each player in the game in order to create a loop in python. In the headers tab in the developer screen we can see that there is a field called “Request URL”. By copying and pasting this link into the browser we can see that the information from the json file is displayed.



```
{"transfers_out": 264561, "code": 15337, "event_total": 2, "last_season_points": 0, "squad_number": null, "news_updated": null, "event_cost": 60, "news_added": "2013-
12-26T19:31:16 UTC+0000", "web_name": "Howard", "in_dreamteam": false, "team_code": 15337, "id": 121, "shirt_image_url":
"http://cdn.ismfg.net/static/plfpl/img/shirts/shirt_6_1.png", "first_name": "Tim", "transfers_out_event": 11035, "element_type_id": 1, "max_cost": 61, "event_explain":
[["Saves", 4, 1], ["Minutes played", 90, 2], ["Goals conceded", 2, -1]], "selected": 288813, "min_cost": 55, "fixtures": {"all": [["03 May 17:30", "Gameweek 37", "Man
City (H)"], ["11 May 15:00", "Gameweek 38", "Hull City (A)"]], "summary": [[37, "MCI (H)", "03 May 17:30"], [38, "HUL (A)", "11 May 15:00"]]}, "season_history":
[["2006/07", 3240, 0, 0, 0, 29, 0, 1, 0, 1, 0, 138, 3, 0, 0, 55, 164], ["2007/08", 3240, 0, 1, 0, 30, 0, 0, 0, 1, 0, 95, 0, 0, 0, 52, 147], ["2008/09", 3420, 0, 0, 0,
37, 0, 0, 0, 0, 0, 116, 0, 0, 0, 52, 159], ["2009/10", 3420, 0, 0, 0, 49, 0, 2, 0, 2, 0, 105, 3, 0, 0, 56, 137], ["2010/11", 3420, 0, 0, 9, 45, 0, 0, 0, 3, 0, 108, 3,
0, 0, 53, 124], ["2011/12", 3420, 1, 0, 12, 40, 0, 2, 0, 1, 0, 88, 3, 428, 0, 54, 151], ["2012/13", 3240, 0, 0, 10, 39, 1, 0, 0, 0, 0, 98, 0, 409, 0, 53, 117]],
"total_points": 151, "type_name": "Goalkeeper", "team_name": "Everton", "status": "a", "added": "2013-07-15T13:21:13 UTC+0000", "form": 5.6, "shirt_mobile_image_url":
"http://cdn.ismfg.net/static/plfpl/img/shirts/mobile/shirt_6_1.png", "current_fixture": "Southampton (A)", "now_cost": 61, "points_per_game": 4.3, "transfers_in":
408567, "news": "", "original_cost": 55, "event_points": 2, "news_return": "2013-12-29T14:01:15 UTC+0000", "fixture_history": {"all": [["17 Aug 15:00", 1, "NOR(A) 2-2",
90, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 5, 3, 0, 55, 1], ["24 Aug 15:00", 2, "WBA(H) 0-0", 90, 0, 0, 1, 0, 0, 0, 0, 0, 2, 0, 8, 7, -5081, 55, 6], ["31 Aug 15:00", 3,
"CAR(A) 0-0", 90, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 7, 7, -176, 55, 6], ["14 Sep 17:30", 4, "CHE(H) 1-0", 90, 0, 0, 1, 0, 0, 0, 0, 0, 5, 0, 18, 8, -5211, 55, 7], ["21
Sep 15:00", 5, "WHU(A) 3-2", 90, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 12, 3, 723, 55, 1], ["30 Sep 20:00", 6, "NEW(H) 3-2", 90, 0, 1, 0, 2, 0, 0, 0, 0, 0, 4, 0, 23, 8,
-2385, 55, 5], ["05 Oct 12:45", 7, "MCI(A) 1-3", 90, 0, 0, 0, 3, 1, 1, 0, 1, 0, 4, 0, 5, 5, -2808, 55, 4], ["19 Oct 15:00", 8, "HUL(H) 2-1", 90, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 12, 3, 418, 55, 2], ["26 Oct 15:00", 9, "AVL(A) 2-0", 90, 0, 0, 1, 0, 0, 1, 0, 1, 0, 3, 3, 15, 12, -2296, 55, 14], ["03 Nov 13:30", 10, "TOT(H) 0-0", 90, 0, 0,
1, 0, 0, 0, 0, 1, 0, 6, 0, 12, 8, 9023, 55, 7], ["09 Nov 15:00", 11, "CRY(A) 0-0", 90, 0, 0, 1, 0, 0, 0, 0, 0, 2, 0, 8, 7, 23393, 55, 6], ["23 Nov 12:45", 12,
"LIV(H) 3-3", 90, 0, 0, 0, 3, 0, 0, 0, 0, 0, 2, 0, 8, 2, 15868, 55, 1], ["30 Nov 15:00", 13, "STK(H) 4-0", 90, 0, 0, 1, 0, 0, 0, 0, 0, 0, 3, 0, 16, 8, 14825, 56, 7],
["04 Dec 19:45", 14, "MUN(A) 1-0", 90, 0, 0, 1, 0, 0, 0, 0, 0, 6, 0, 19, 9, 11492, 56, 8], ["08 Dec 16:00", 15, "ARS(A) 1-1", 90, 0, 0, 0, 1, 0, 0, 0, 1, 0, 4, 0, 8,
3, 22966, 57, 2], ["14 Dec 15:00", 16, "FUL(H) 4-1", 90, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 14, 2, 69663, 58, 2], ["22 Dec 16:00", 17, "SWA(A) 2-1", 90, 0, 0, 0, 1, 0, 0,
0, 0, 0, 3, 0, 17, 4, 15394, 59, 3], ["26 Dec 15:00", 18, "SUN(H) 0-1", 22, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, -2, -3, 27405, 59, -2], ["29 Dec 13:30", 19, "SOU(H) 2-1",
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -82314, 59, 0], ["01 Jan 15:00", 20, "STK(A) 1-1", 90, 0, 0, 0, 1, 0, 0, 0, 0, 0, 3, 0, 9, 4, -4372, 59, 3], ["11 Jan 15:00",
21, "NOR(H) 2-0", 90, 0, 0, 1, 0, 0, 0, 0, 0, 0, 3, 0, 16, 8, 1240, 59, 7], ["20 Jan 20:00", 22, "WBA(A) 1-1", 90, 0, 0, 0, 1, 0, 0, 0, 0, 0, 5, 0, 11, 4, 3545, 59, 3],
["28 Jan 20:00", 23, "LIV(A) 0-4", 90, 0, 0, 0, 4, 0, 0, 0, 0, 0, 5, 0, 9, 1, -6554, 59, 1], ["01 Feb 15:00", 24, "AVL(H) 2-1", 90, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 12,
2, -4185, 59, 2], ["09 Feb 13:30", 25, "TOT(A) 0-1", 90, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 4, 2, -4632, 59, 2], ["22 Feb 12:45", 27, "CHE(A) 0-1", 90, 0, 0, 0, 1, 0, 0,
0, 0, 0, 7, 0, 9, 5, -9321, 59, 4], ["01 Mar 15:00", 28, "WHU(H) 1-0", 90, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 14, 7, 1840, 59, 6], ["15 Mar 15:00", 30, "CAR(H) 2-1", 90,
0, 0, 0, 1, 0, 0, 0, 0, 0, 2, 0, 14, 3, 9345, 59, 2], ["22 Mar 15:00", 31, "SWA(H) 3-2", 90, 0, 0, 0, 2, 0, 0, 0, 0, 0, 5, 0, 18, 4, 11435, 59, 2], ["25 Mar 19:45", 31,
"NEW(A) 3-0", 90, 0, 0, 1, 0, 0, 0, 0, 0, 3, 0, 15, 8, 11435, 59, 7], ["30 Mar 13:30", 32, "FUL(A) 3-1", 90, 0, 0, 0, 1, 0, 0, 0, 0, 0, 7, 0, 20, 6, 6550, 59, 4],
["06 Apr 13:30", 33, "ARS(H) 3-0", 90, 0, 0, 1, 0, 0, 0, 0, 0, 0, 5, 0, 18, 8, -1601, 59, 7], ["12 Apr 15:07", 34, "SUN(A) 1-0", 90, 0, 0, 1, 0, 0, 0, 0, 0, 0, 5, 3,
18, 8, 31588, 60, 10], ["16 Apr 19:45", 34, "CRY(H) 2-3", 90, 0, 0, 0, 3, 0, 0, 0, 0, 0, 1, 0, 3, 2, 31588, 60, 1], ["20 Apr 16:10", 35, "MUN(H) 2-0", 90, 0, 0, 1, 0,
0, 0, 0, 0, 0, 3, 1, 16, 8, -2573, 60, 8], ["26 Apr 12:45", 36, "SOU(A) 0-2", 90, 0, 0, 0, 2, 0, 0, 0, 0, 0, 4, 0, 6, 4, 14485, 60, 2]], "summary": [[34, "CRY (H)", 1],
[35, "MUN (H)", 8], [36, "SOU (A)", 2]]}, "next_fixture": "Man City (H)", "transfers_in_event": 2501, "selected_by": "9.0", "team_id": 6, "second_name": "Howard",
"photo_mobile_url": "http://cdn.ismfg.net/static/plfpl/img/shirts/photos/15337.jpg"}
```

Fig 3.12: JSON File

After some investigation it was concluded that by altering the last number in this link we could retrieve each players scoring and history data. The id in this link relates to the individual player's id in the game which begins at 1 for the first player i.e. the Arsenal goalkeeper and ends with the number of the last player to have come into the system during the year. This could be a player who has transferred into a club at some stage of the season or perhaps a youth team player brought into the senior squad.

Now that the structure of the website and the location of the required data was known, the next activity was to create the python script which would scrape all of the data for us.

3.2.2 Python Web Scraping JSON Script

The first thing that needed to be created was a connection between python and the website. The inbuilt library in python called urllib was used to achieve this. This module provides a high-level interface for fetching data across the World Wide Web and the urlopen() function accepts URLs as inputs for creating connections and opening network objects (Python, 2014).

The library was imported into the python session using the command

import urllib

and the connection was made to the API which contains the json file using:

htmltext = urllib.urlopen("http://fantasy.premierleague.com/web/api/elements/" + str(i) + "/")

The str(i) section of the URL here enabled us to create a variable "i" which we could loop in order to create a connection with the file relating to each player. We needed first to find what range was required for i so that all players in the game would be included. An initial estimate range of 1- 1,000 was used and this was refined as the script was developed and tested. For initial test purposes, the value of i was just taken as 1. Once the process was functional for a single player then the looping system would be implemented for all players.

In order to interpret the json file the package "json" was imported into the session. JSON (JavaScript Object Notation) is a compact, text based format for computers to exchange data and is once loaded into Python just like a dictionary. JSON data structures map directly to Python data types, which makes this a powerful tool for directly accessing data without having to write any XML parsing code (Python for Beginners, 2014). A variable called "data" was created which was the entire json file handled using the json library using the function *json.load()*:

data = json.load(htmltext)

Once this was defined, the required fields could be chosen from the file as required. In order to pull back the players entire scoring history the command:

scoredata = data["fixture_history"]["all"]

Other data such as the player name, value, position, team name and percentage of managers selected by were extracted in the same manner and stored as variables.

In order to view the output of the results, a text file was created in the script and all of the data was appended to the text file using comma separated values (csv). The file was created at the start of the script using:

myfile = open("player_history.txt", "w")

and each of the variables were appended to the file for each line in the data. The “w” at the end of this command implies that the file is writable meaning every time the script is run the new data will replace the old data.

myfile.write(playerdata + "," + teamname + "," + position + "," + selected + "," + str(price) + ',' + str(i) + "\n")

3.2.2.1 Create the Loop

Once the system was in place for an individual player it was necessary to introduce a loop into the script which could iterate through all of the players in the game and append their data to the master file. As mentioned in the previous section, a variable called “i” was used to store the player id and increment by 1 as the loop progresses.

The basic structure of the loop is shown below in pseudo code:

```
1. Create writable text file
2. Close the text file
3. Set i = 1
4. Create Loop (while i < 1000):
    a. Open a connection to the API URL htmltext =
       urllib.urlopen("http://fantasy.premierleague.com/web/api/elements/" + str(i) + "/")
    b. Create the json object using json.load(htlmtext).
    c. Extract the scoring history, player name, player team, player value and percentage
       of managers selected by.
    d. Perform some data cleansing on the file (see Appendix E for code )
    e. Append all of the different variables to the file using comma separated values.
    f. Increment the value of i by 1 (i +=1)
5. Print “Process complete” to the terminal.
```

This loop will run until the value of i reaches 1000 at which point it will stop and the script will be complete. The output of the loop should be a text file with each value separated by a comma and each row on a new line of the file (“\n”).

3.2.2.2 Testing the Script

When the script was run initially it was noticed that errors were occurring and the script was breaking at certain points along its execution. In order to discover the exact points at which the script was breaking, a test file was incorporated into the script which allowed us to view the exact points in the script and exact values of i in the loop which were causing the issues.

At the start of the script, a new file was created:

errfile = open("errfile.txt", "w")

The purpose of this file was to capture information on the location of the loop at which the code was failing. In order to write to this file an error handling process was added to the script. Error handling in python is done using a try-except block statement.

The try-except statement block works as follows.

1. First, the *try clause* (the statement(s) between the try and except keywords) is executed.
2. If no exception occurs, the except clause is skipped and execution of the try statement is finished.
3. If an exception occurs during execution of the try clause, the rest of the clause is skipped. Then if its type matches the exception named after the except keyword, the except clause is executed, and then execution continues after the try statement.

For our file the try-except block statement worked as follows:

1. Begin Loop
2. Try to execute the code within the loop for each individual player
3. If no exception occurs skip the except clause and move onto the next iteration of the loop
4. If an exception does occur skip the execution of the try block and execute the except block.
5. Within the except block open the errfile.txt file.
6. Write the value of i to the file so it can be investigated manually.
7. Move onto next iteration of the loop.

The final output of this code was a file containing the list of all player ids which caused the script to fail. Once this list was available it was then possible to manually enter the player ids into the API to check for any common features that may be causing the failure of the program.

It was discovered that there were two main reasons for the failure of the script. The first reason was that the range of the loop was too large meaning that we were accounting for more players than were actually in the game. When the program encountered a player id which did not exist and try to access the API url for this player, an error was thrown as the URL did not exist. This error was fixed by refining the loop to the maximum number of players in the game. However, this meant that if any new players were added in the mean time, they would not be included in the program. In order to maintain the integrity of the program, the try-except block was maintained in the program and a float of 10 above the maximum number of players was built in which allowed for any new players added at any stage. Even if no new players were added, the code would still execute due to the presence of the try-except block.

The other reason for the failure of the program was the presence of certain characters in the names of some of the foreign players in the game. For example, Andre Shürrle has the special character ü in the name. The default encoding of files in python is ASCII which only defines unaccented characters (Python, 2014). In order to overcome this situation the encoding of the file was changed to utf-8 (Unicode Transformation Format 8) which can handle any Unicode code point. In order to do this the library "io" was imported into the session and the default encoding of the file was changed to UTF-8:

import io

myfile = io.open("player_history.txt", "a", encoding='utf8')

When this change was made the script ran without any errors and the file was saved off with all of the required data.

3.2.3 Python Web Scraping HTML Script

While the vast majority of the data required for the analysis was extracted from the JSON files in the previous section, some other supplementary information was required which was not available in the JSON file. In order to extract the league table and fixture details from the website it was necessary to scrape the background HTML from the website. The library used to carry this out in python was called BeautifulSoup.

Beautiful Soup (BS) is a Python library for pulling data out of HTML and XML files. It works with the parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work (Crummy, 2013). In order to begin using the BS library it must first be downloaded. It can be downloaded as a tar.gz file and extracted using the 7zip software or you can install it with with pip install BeautifulSoup or easy_install BeautifulSoup from the command line.

The library is imported into the session using:

from bs4 import BeautifulSoup

3.2.3.1 Web Scraping Fixtures and Results

A text file created to store the fixtures and results information from the script:

fix_file = open('fixtures.txt', 'w')

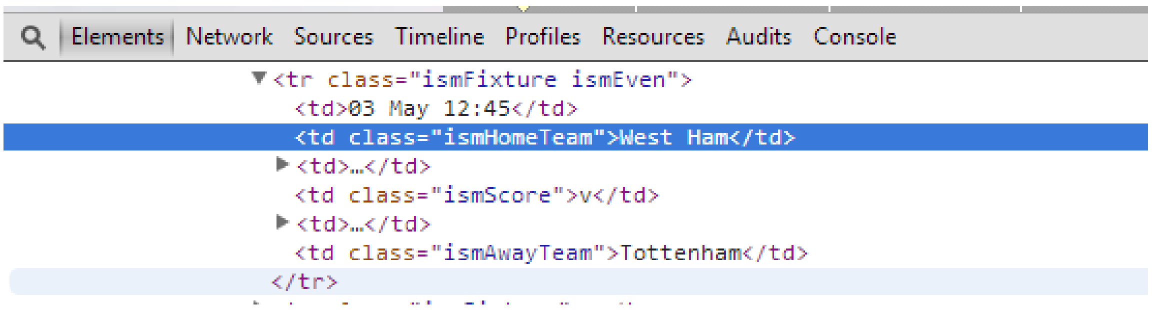
The connection was made with the website again using the urllib.open() function. In order to scrape the fixtures information from the website a loop was created which iterated through each of the 38 game weeks and extracted every result and upcoming fixture left to be played. The html code was read in using:

html = urllib.urlopen(myurl).read(), where myurl = "http://fantasy.premierleague.com/fixtures/" + str(i)

The loop used for this was a while loop with a range of 1 – 38 for the value of the variable "week". The BeautifulSoup function BeautifulSoup () was then used to parse all of the HTML text:

soup = BeautifulSoup(html)

Once this was complete we needed to find what HTML tags contained the information we were looking for. Once again, the developer screen in chrome was used by right clicking on the data in the website and finding the tag in the HTML that was required.



```
▼<tr class="ismFixture ismEven">
   <td>03 May 12:45</td>
   <td class="ismHomeTeam">West Ham</td>
  ▶<td>…</td>
   <td class="ismScore">v</td>
  ▶<td>…</td>
   <td class="ismAwayTeam">Tottenham</td>
 </tr>
```

Fig 3.12: Background HTML code for fixture details

When the required tags were identified, it was a matter of writing the code which would pull out the information from these tags for us.

hometeam = soup.findAll("td", {"class":"ismHomeTeam"})

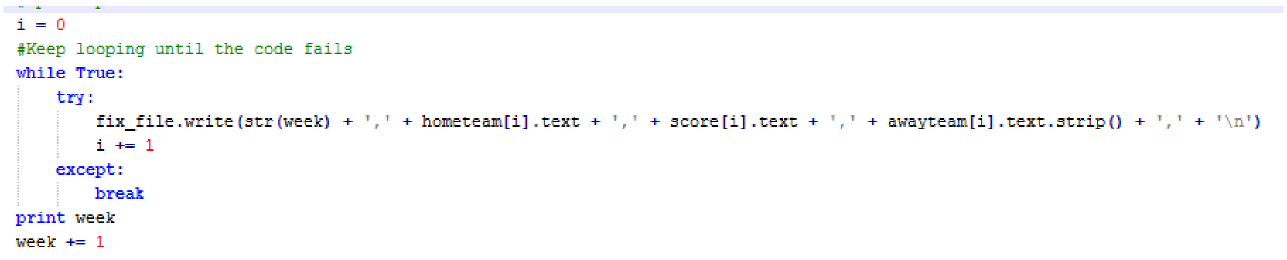
awayteam = soup.findAll("td", {"class":"ismAwayTeam"})

score = soup.findAll("td", {"class":"ismScore"})

The above code pulls out the home team (tag = “td”, class = “ismHomeTeam”), away team (tag = “td”, class = “ismAwayTeam”) and the score in the fixture (tag = “td”, class = “ismScore”). All of this information is then appended to the text file using:

fix_file.write(str(week) + ',' + hometeam[i].text + ',' + score[i].text + ',' + awayteam[i].text.strip() + ',' + '\n'), where i is the number of fixtures in that game week.

As different game weeks have different numbers of fixtures (some teams may play twice in a game week and some teams may not play at all) the value of i varies from game week to game week. To counteract this, instead of setting a range for the value of i as we had before, a loop was created which continued executing until it failed:



```
i = 0
#Keep looping until the code fails
while True:
    try:
        fix_file.write(str(week) + ',' + hometeam[i].text + ',' + score[i].text + ',' + awayteam[i].text.strip() + ',' + '\n')
        i += 1
    except:
        break
print week
week += 1
```

Fig 3.13: Python Looping Code

This created a line in the text file for each fixture for each game week with the name of the home team, the away team and the score in the fixture.

3.2.3.1 Web Scraping League Table

Scraping the league table was done in a similar manner to the fixtures scraping. The URL for the website was defined and a connection made using the urllib.open() function:

html = urllib.urlopen(base).read(), where base = "http://fantasy.premierleague.com/transfers/"

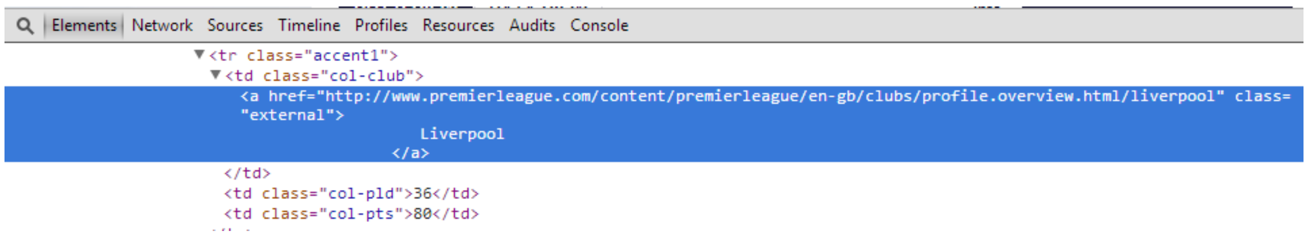
The BeautifulSoup() function was used to parse all of the HTML of the website:

soup = BeautifulSoup(html)

A text file was created which would store all of the information scraped by the program.

league_table = open("league_table.txt", "w")

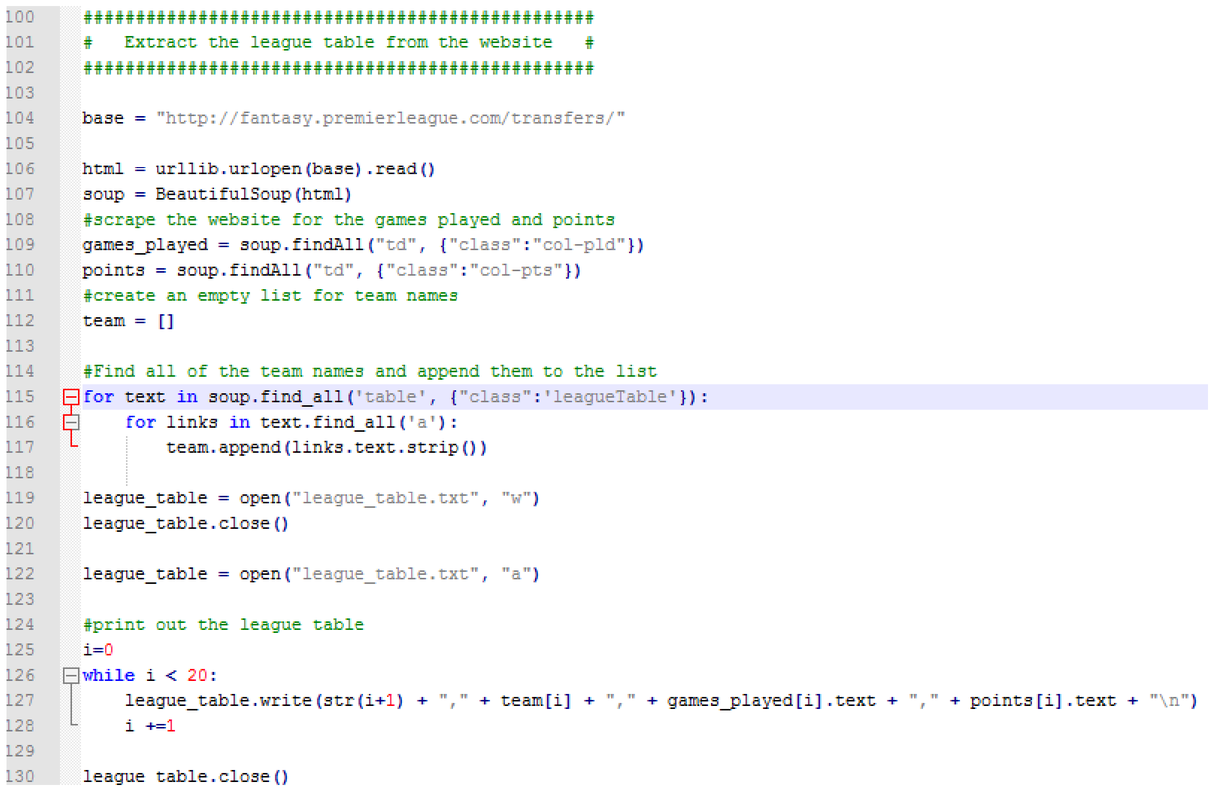
The tags with the required information in the HTML were identified.



```
▼<tr class="accent1">
  ▼<td class="col-club">
      <a href="http://www.premierleague.com/content/premierleague/en-gb/clubs/profile.overview.html/liverpool" class=
      "external">
                          Liverpool
                      </a>
    </td>
    <td class="col-pld">36</td>
    <td class="col-pts">80</td>
```

Fig 3.14: Background League Table code

And the information scraped from them:



```
#################################################
#   Extract the league table from the website   #
#################################################

base = "http://fantasy.premierleague.com/transfers/"

html = urllib.urlopen(base).read()
soup = BeautifulSoup(html)
#scrape the website for the games played and points
games_played = soup.findAll("td", {"class":"col-pld"})
points = soup.findAll("td", {"class":"col-pts"})
#create an empty list for team names
team = []

#Find all of the team names and append them to the list
for text in soup.find_all('table', {"class":'leagueTable'}):
    for links in text.find_all('a'):
        team.append(links.text.strip())

league_table = open("league_table.txt", "w")
league_table.close()

league_table = open("league_table.txt", "a")

#print out the league table
i=0
while i < 20:
    league_table.write(str(i+1) + "," + team[i] + "," + games_played[i].text + "," + points[i].text + "\n")
    i +=1

league_table.close()
```

Fig 3.15: program to scrape league table

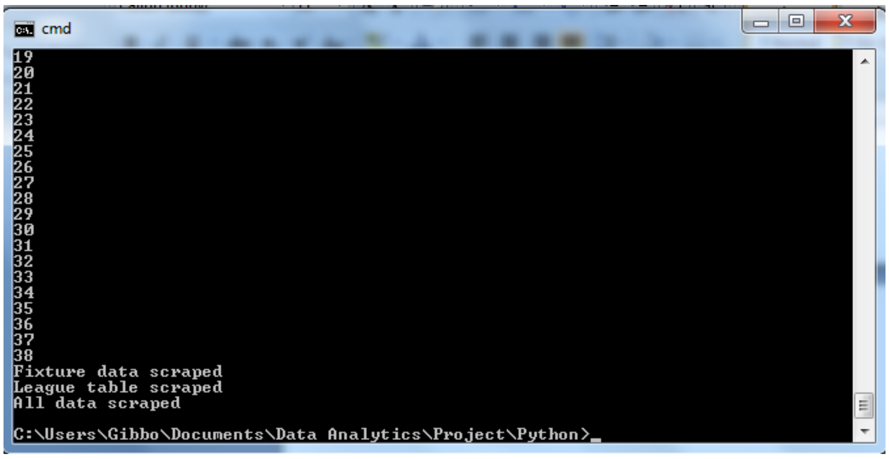
The output of this program was a csv text file which has the columns, table position, team name, games played and points. As there are only 20 teams in the league, the loop for this was simpler than the previous program we had a definite start and end value for our iterator, i.

3.3 Running the System

All three scripts were saved to a single, executable python file called master.py. This file was executed from the command line using the command:

“Python master.py”

As the code iterates through each of the steps a message was printed to the terminal which gives details on the progress of the program letting the user know exactly the section of code the program is executing at any time.



```
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
Fixture data scraped
League table scraped
All data scraped

C:\Users\Gibbo\Documents\Data Analytics\Project\Python>_
```

Fig 3.16: Output to terminal

All of the code for the scraping activities can be found in Appendix E.

3.4 Copyright Information

The scraping of data from websites on the internet is something of a legal grey area with the owners of each website having different rules for downloading content. The FPL website gives the following guidelines to avoiding copyright infringement:

- You may download and print material from the Website as is reasonable for your own private and personal use.
- You may forward such material from the Website to other people for their private and personal use too provided you credit the FPL as its source and add the Website address: www.premierleague.com. You must draw their attention to these terms which also apply to them (Premier League, 2014).

In this case it was interpreted that any information scraped from the website was legal as long as it would not be used for commercial purposes. In this case it is not as it is required for an academic project. The dashboard which will be created to distribute the information will have a link to the home page of the website as specified above.

4. Mysql Database

4.1 Introduction

Mysql is the second most widely used open source relational database management system (RDBMS) in the world (DB Engines, 2013). It is developed by Oracle and released originally in 1995. It uses the SQL (Structured Query Language) and is implemented in C and C++. The main command line tool is the command line client which allows the user to input code directly into the terminal or to run previously saved SQL scripts.

4.2 Purpose

Mysql was used in this project to create a database of all of the information which was scraped from the FPL website in the form of a number of different tables. Using Mysql a lot of data manipulation was carried out on the underlying data in order to prepare for the data load into the Qlikview environment.

The Mysql database was act as a bridge between the raw data taken from the website and the final dashboard to be presented to the end user. The tables that were created in Mysql could be easily loaded into qlikview via ODBC (Open Database Connectivity).

4.3 Download

MySQL Community Edition is a freely downloadable version of Mysql which is supported by an active community of open source developers and enthusiasts. It can be downloaded from the Mysql website (MySQL, 2014) across any platform. The user must first create an account with Mysql before beginning the download but once this is complete the download can commence. Once the files are downloaded to the C drive of the machine the user can begin loading data, creating databases and tables and avail of all the functionality Mysql has to offer.

4.4 Create Database and Tables

A database was created for the sole purpose of storing the tables relating to the project. The database was called PROJECT and it was created using the following command:

```
create database PROJECT;

use PROJECT;
```

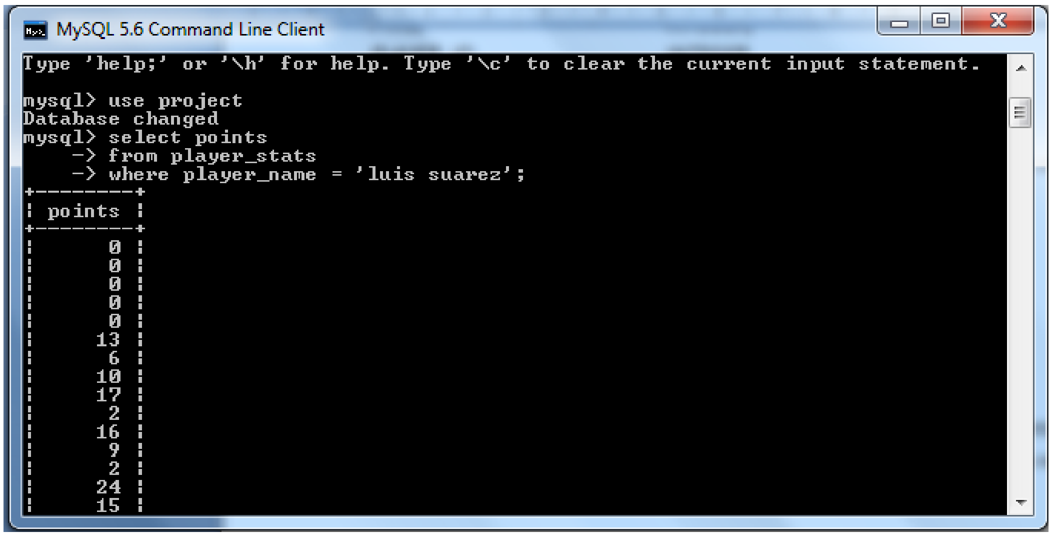
Once the database was initialised using the “use” function, the tables which were to be used for holding the data were created. The main table was called PLAYER_STATS and had the following fields and datatypes:

```
CREATE TABLE PLAYER_STATS(

GAME_DTE                    VARCHAR(30),
GAMEWEEK                    INTEGER,
OPPOSITION                  VARCHAR(30),
MINS_PLYD                   INTEGER,
GOALS_SCORED                INTEGER,
ASSISTS                     INTEGER,
CLEAN_SHEET          INTEGER,
GOALS_CONCEDED       INTEGER,
OWN_GOALS            INTEGER,
PENALTIES_SAVED      INTEGER,
PENALTIES_MISSED     INTEGER,
YELLOW_CARDS         INTEGER,
RED_CARDS            INTEGER,
SAVES                INTEGER,
BONUS                INTEGER,
EA_PPI               INTEGER,
BONUS_POINTS_SYS     INTEGER,
NET_TRANSFERS        INTEGER,
PLAYER_VALUE         INTEGER,
POINTS               INTEGER,
PLAYER_NAME          VARCHAR(50),
TEAM_NAME            VARCHAR(30),
POSITION             VARCHAR(30),
SELECTED_BY          VARCHAR(5),
PRICE                INTEGER,
PLAYER_ID            INTEGER
)
;
```

In order to load the scraped data from the text file into the newly created table the following command was used:

```
LOAD DATA LOCAL INFILE "C:/Users/Gibbo/Documents/Data
Analytics/Project/Python/player_history.txt"
INTO TABLE PLAYER_STATS
COLUMNS TERMINATED BY ','
;
```

This command takes the text file player_history.txt and loads the data into the table PLAYER_STATS. Once this was complete it was then possible to select from the table as required.

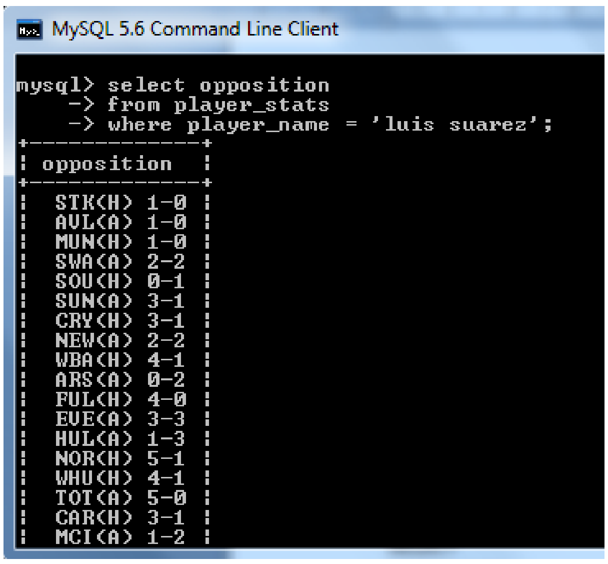


```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use project
Database changed
mysql> select points
    -> from player_stats
    -> where player_name = 'luis suarez';
+--------+
| points |
+--------+
|      0 |
|      0 |
|      0 |
|      0 |
|      0 |
|     13 |
|      6 |
|     10 |
|     17 |
|      2 |
|     16 |
|      9 |
|      2 |
|     24 |
|     15 |
```

Fig 4.1: MySQL Terminal

4.5 Data Manipulation and Cleansing

Some of the fields in this table required some manipulation in order for them to be useful in the final analysis. For example, the OPPOSITION field in the table is a character field which gives an abbreviated name of the opposition team, whether it was a home or away match and the final score.



```
mysql> select opposition
    -> from player_stats
    -> where player_name = 'luis suarez';
+-------------+
| opposition  |
+-------------+
|   STK(H) 1-0 |
|   AVL(A) 1-0 |
|   MUN(H) 1-0 |
|   SWA(A) 2-2 |
|   SOU(H) 0-1 |
|   SUN(A) 3-1 |
|   CRY(H) 3-1 |
|   NEW(A) 2-2 |
|   WBA(H) 4-1 |
|   ARS(A) 0-2 |
|   FUL(H) 4-0 |
|   EVE(A) 3-3 |
|   HUL(A) 1-3 |
|   NOR(H) 5-1 |
|   WHU(H) 4-1 |
|   TOT(A) 5-0 |
|   CAR(H) 3-1 |
|   MCI(A) 1-2 |
```

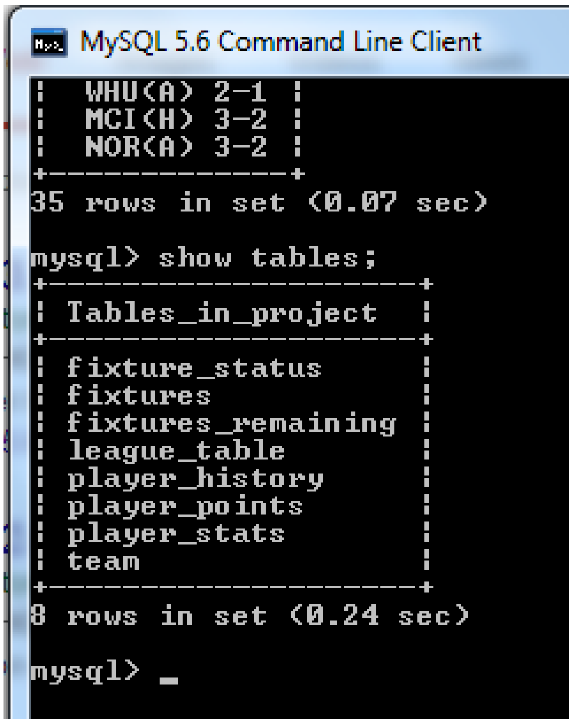
Fig 4.2: Output from query 1

Ideally we would like to have these as three separate fields so some manipulation was required as shown below:

```
CREATE TABLE PLAYER_HISTORY
AS
(
SELECT
GAME_DTE,
GAMEWEEK,
SUBSTRING(TRIM(OPPOSITION), 1, 3) AS OPPOSITION,
SUBSTRING(TRIM(OPPOSITION), 5, 1) AS HOME_AWAY,
SUBSTRING(TRIM(OPPOSITION), 8, 3) AS SCORE,

CASE WHEN SUBSTRING(TRIM(OPPOSITION), 8, 1) > SUBSTRING(TRIM(OPPOSITION), 10, 1) THEN
'WIN'
        WHEN SUBSTRING(TRIM(OPPOSITION), 8, 1) < SUBSTRING(TRIM(OPPOSITION), 10, 1) THEN
'LOSE'
        ELSE 'DRAW'
END AS MATCH_OUTCOME,

MINS_PLYD                   ,
GOALS_SCORED                ,
ASSISTS                    ,
CLEAN_SHEET                ,
GOALS_CONCEDED             ,
OWN_GOALS                  ,
PENALTIES_SAVED            ,
PENALTIES_MISSED           ,
YELLOW_CARDS               ,
RED_CARDS                  ,
SAVES                      ,
BONUS                      ,
EA_PPI                     ,
BONUS_POINTS_SYS           ,
NET_TRANSFERS              ,
PLAYER_VALUE               ,
POINTS                     ,
PLAYER_NAME                ,
TEAM_NAME                  ,
POSITION                   ,
CAST(SELECTED_BY AS DECIMAL(3,1)) AS SELECTED_BY,
PRICE                      ,
PLAYER_ID

FROM PLAYER_STATS
)
;
```

In this instance, the table PLAYER_HISTORY was created by selecting data from the PLAYER_STATS table and manipulating some of the fields into the desired format. The new field OPPOSITION extracts the abbreviated team name from the original OPPOSITION field in PLAYER_STATS. The fields HOME_AWAY and SCORE create two new fields from OPPOSITION which tell us if the team played at home or away and what the score of the match was. The field MATCH_OUTCOME tells us whether the team won, drew or lost the match based on the score while the field SELECTED_BY modifies the original character SELECTED_BY field to a decimal.

A total of 8 tables were created in mysql. These are shown in the console below:



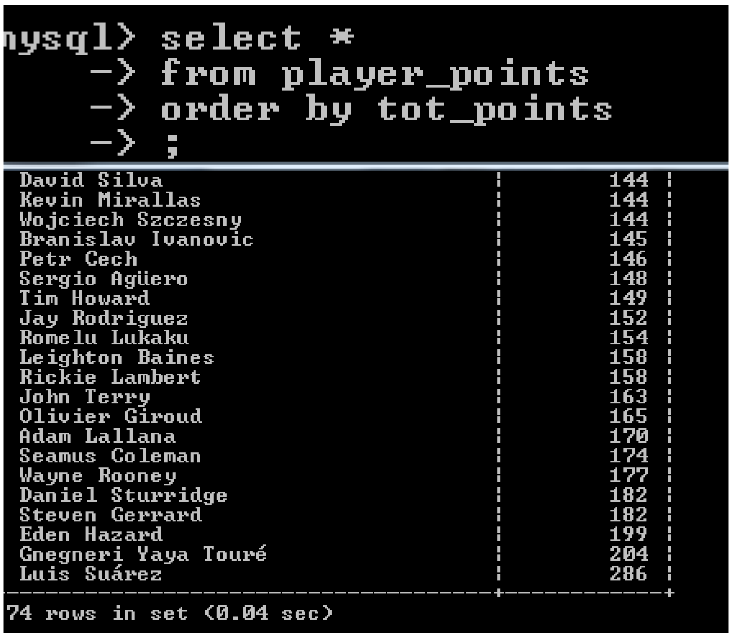
```
MySQL 5.6 Command Line Client
|   WHU(A) 2-1  |
|   MCI(H) 3-2  |
|   NOR(A) 3-2  |
+---------------+
35 rows in set (0.07 sec)

mysql> show tables;
+--------------------+
| Tables_in_project  |
+--------------------+
| fixture_status     |
| fixtures           |
| fixtures_remaining |
| league_table       |
| player_history     |
| player_points      |
| player_stats       |
| team               |
+--------------------+
8 rows in set (0.24 sec)

mysql> _
```

Fig 4.3: Output from query 2

The table TEAM was created as a reference table which mapped the abbreviated name in the PLAYER_HISTORY table to an actual team name. The structure of the table is shown below:

```
CREATE TABLE TEAM(
TEAM_NAME                VARCHAR(30),
TEAM_ABBRV               CHAR(3)
)
;

INSERT INTO TEAM VALUES('Arsenal', 'ARS');
INSERT INTO TEAM VALUES('Crystal Palace', 'CRY');
INSERT INTO TEAM VALUES('Aston Villa', 'AVL');
INSERT INTO TEAM VALUES('Stoke City', 'STK');
INSERT INTO TEAM VALUES('Fulham', 'FUL');
INSERT INTO TEAM VALUES('Cardiff City', 'CAR');
INSERT INTO TEAM VALUES('Chelsea', 'CHE');
INSERT INTO TEAM VALUES('Man Utd', 'MUN');
INSERT INTO TEAM VALUES('Liverpool', 'LIV');
INSERT INTO TEAM VALUES('Everton', 'EVE');
INSERT INTO TEAM VALUES('West Brom', 'WBA');
INSERT INTO TEAM VALUES('Hull City', 'HUL');
INSERT INTO TEAM VALUES('West Ham', 'WHU');
INSERT INTO TEAM VALUES('Sunderland', 'SUN');
INSERT INTO TEAM VALUES('Man City', 'MCI');
INSERT INTO TEAM VALUES('Newcastle', 'NEW');
INSERT INTO TEAM VALUES('Norwich', 'NOR');
INSERT INTO TEAM VALUES('Southampton', 'SOU');
INSERT INTO TEAM VALUES('Swansea', 'SWA');
INSERT INTO TEAM VALUES('Tottenham', 'TOT');
```

The table PLAYER_POINTS provides a summary of the total points of all players in the game:



```
nysql> select *
    -> from player_points
    -> order by tot_points
    -> ;
 David Silva                          |        144 |
 Kevin Mirallas                       |        144 |
 Wojciech Szczesny                    |        144 |
 Branislav Ivanovic                   |        145 |
 Petr Cech                            |        146 |
 Sergio Agüero                        |        148 |
 Tim Howard                           |        149 |
 Jay Rodriguez                        |        152 |
 Romelu Lukaku                        |        154 |
 Leighton Baines                      |        158 |
 Rickie Lambert                       |        158 |
 John Terry                           |        163 |
 Olivier Giroud                       |        165 |
 Adam Lallana                         |        170 |
 Seamus Coleman                       |        174 |
 Wayne Rooney                         |        177 |
 Daniel Sturridge                     |        182 |
 Steven Gerrard                       |        182 |
 Eden Hazard                          |        199 |
 Gnegneri Yaya Touré                  |        204 |
 Luis Suárez                          |        286 |
--------------------------------------+------------+
74 rows in set (0.04 sec)
```

Fig 4.4: Output from query 3

The tables LEAGUE_TABLE and FIXTURES were created by loading all of the data from the text files league_table.txt and fixtures.txt:

```
CREATE TABLE LEAGUE_TABLE
(
POSITION                    INTEGER,
TEAM_NAME                   VARCHAR(30),
GAMES_PLAYED                INTEGER,
POINTS                      INTEGER
)
;

LOAD DATA LOCAL INFILE
"C:/Users/Gibbo/Documents/DataAnalytics/Project/Python/league_table.txt"
INTO TABLE LEAGUE_TABLE
COLUMNS TERMINATED BY ','
;

CREATE TABLE FIXTURES
(
GAMEWEEK                    INTEGER,
HOME_TEAM                   VARCHAR(30),
SCORE                       VARCHAR(10),
AWAY_TEAM                   VARCHAR(30)
)
;
```

```
LOAD DATA LOCAL INFILE "C:/Users/Gibbo/Documents/Data Analytics/Project/Python/fixtures.txt"
INTO TABLE FIXTURES
COLUMNS TERMINATED BY ','
;
```

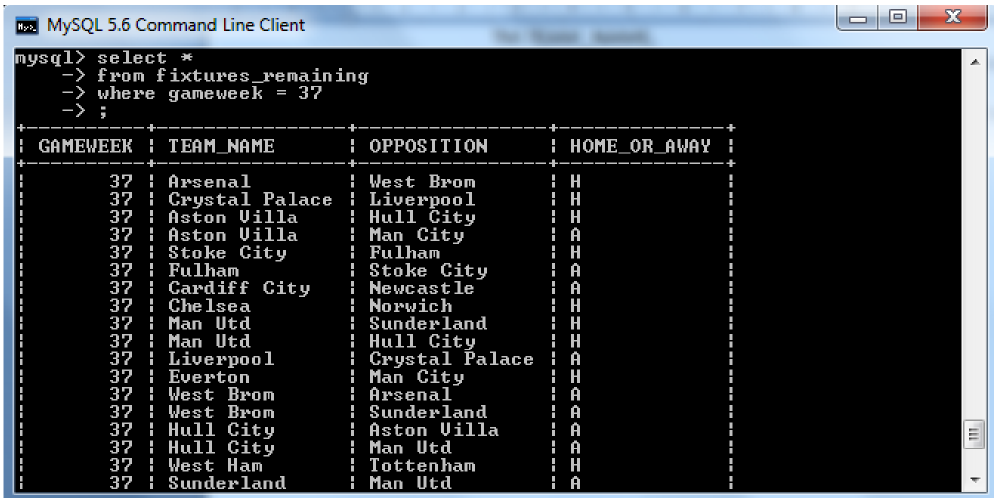
The table FIXTURE_STATUS was created to modify the FIXTURE table to contain an identifier which indicates if the match is complete or is still to be played:

```
CREATE TABLE FIXTURE_STATUS
AS
(
SELECT GAMEWEEK,
                    HOME_TEAM,
                    SCORE,
                    AWAY_TEAM,
                    CASE WHEN SCORE LIKE ('%0%')
                              OR SCORE LIKE ('%1%')
                              OR SCORE LIKE ('%2%')
                              OR SCORE LIKE ('%3%')
                              OR SCORE LIKE ('%4%')
                              OR SCORE LIKE ('%5%')
                              OR SCORE LIKE ('%6%')
                              OR SCORE LIKE ('%7%')
                              OR SCORE LIKE ('%8%')
                              OR SCORE LIKE ('%9%') THEN 'RESULT' ELSE 'TBP' END AS STATUS

FROM FIXTURES
)
;
```

The table FIXTURES_REMAINING was created to provide an overall view of the remaining fixtures for each team. The code is shown below:

```
CREATE TABLE FIXTURES_REMAINING
AS
(
SELECT FIX.GAMEWEEK,
                    TM.TEAM_NAME,
                    CASE WHEN TM.TEAM_NAME = FIX.HOME_TEAM THEN FIX.AWAY_TEAM ELSE
FIX.HOME_TEAM END AS OPPOSITION,
                    CASE WHEN TM.TEAM_NAME = FIX.HOME_TEAM THEN 'H' ELSE 'A' END AS
HOME_OR_AWAY



FROM FIXTURES                                           FIX

LEFT JOIN TEAM                                          TM
ON TM.TEAM_NAME = FIX.HOME_TEAM
OR TM.TEAM_NAME = FIX.AWAY_TEAM

WHERE SCORE LIKE '%V%'
)
;
```

This table is created by joining to the FIXTURES table to the TEAM table on HOME_TEAM OR AWAY_TEAM. The purpose of this is to create a single row per team per game week remaining and show all the fixtures that team has left. For each game remaining, there will be two rows in the table, one for the away team and one for the home team.



```
mysql> select *
    -> from fixtures_remaining
    -> where gameweek = 37
    -> ;
+----------+----------------+----------------+--------------+
| GAMEWEEK | TEAM_NAME      | OPPOSITION     | HOME_OR_AWAY |
+----------+----------------+----------------+--------------+
|       37 | Arsenal        | West Brom      | H            |
|       37 | Crystal Palace | Liverpool      | H            |
|       37 | Aston Villa    | Hull City      | H            |
|       37 | Aston Villa    | Man City       | A            |
|       37 | Stoke City     | Fulham         | H            |
|       37 | Fulham         | Stoke City     | A            |
|       37 | Cardiff City   | Newcastle      | A            |
|       37 | Chelsea        | Norwich        | H            |
|       37 | Man Utd        | Sunderland     | H            |
|       37 | Man Utd        | Hull City      | H            |
|       37 | Liverpool      | Crystal Palace | A            |
|       37 | Everton        | Man City       | H            |
|       37 | West Brom      | Arsenal        | A            |
|       37 | West Brom      | Sunderland     | A            |
|       37 | Hull City      | Aston Villa    | A            |
|       37 | Hull City      | Man Utd        | A            |
|       37 | West Ham       | Tottenham      | H            |
|       37 | Sunderland     | Man Utd        | A            |
```

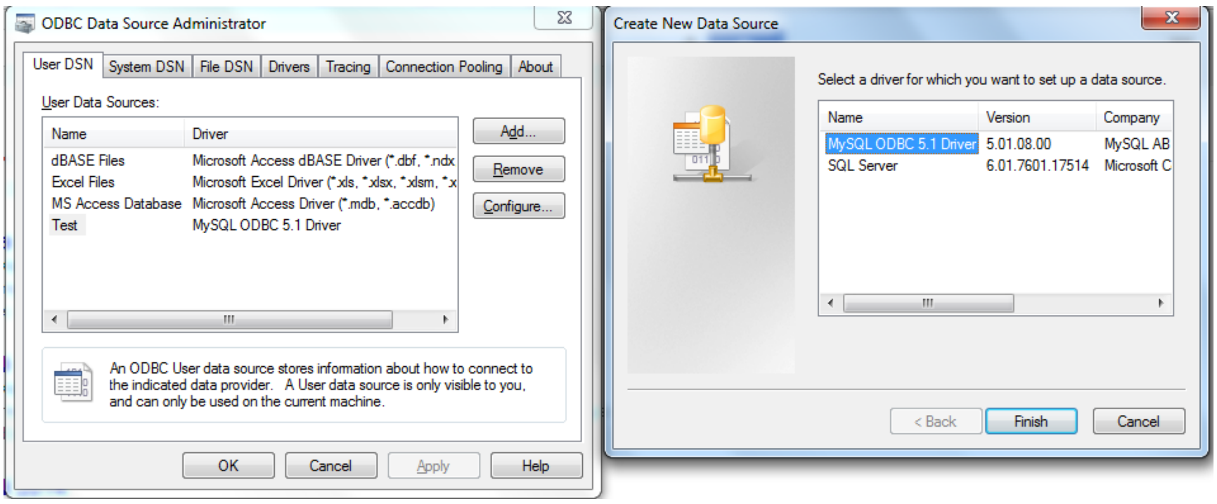
Fig 4.5: Output from query 4

In this table we can see where teams can have multiple games in a single game week. For instance, Man Utd are at home to Sunderland and Hull City in game week 37. The rest of the code for the mySQL database can be viewed in Appendix F.

4.6 ODBC Driver

In order to connect the Mysql database to the qlikview application an ODBC connection must be made between the two. The ODBC driver can be downloaded from http://dev.mysql.com/downloads/connector/odbc/ . This is a standardized database driver for Windows, Linux, Mac OS X, and Unix platforms (MySQL, 2014). The installer for Windows was downloaded and the default settings maintained throughout the process.

Once the download is complete, the ODBC connection was configured. The ODBC Data Sources was opened in Administrative tools in the Control Panel and a new Data Source Name (DSN) added.



User Data Sources:

Name	Driver
dBASE Files	Microsoft Access dBASE Driver (*.dbf, *.ndx
Excel Files	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.x
MS Access Database	Microsoft Access Driver (*.mdb, *.accdb)
Test	MySQL ODBC 5.1 Driver

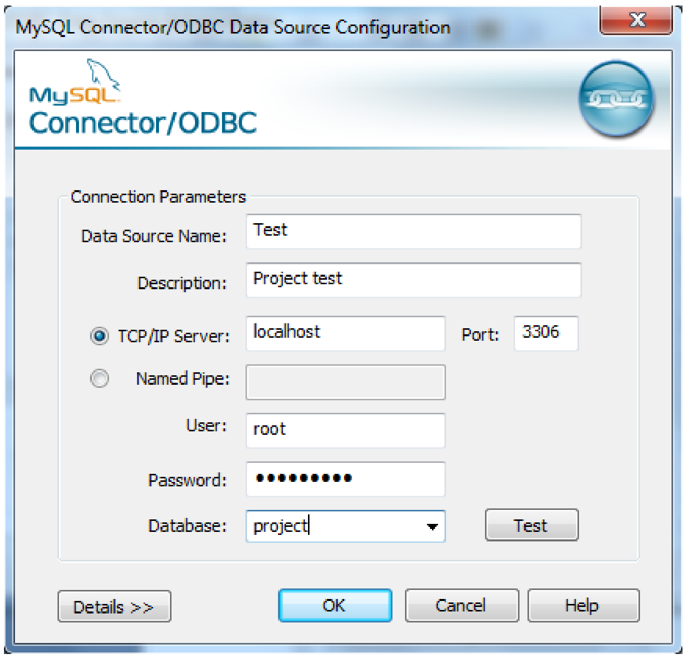
An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

Select a driver for which you want to set up a data source.

Name	Version	Company
MySQL ODBC 5.1 Driver	5.01.08.00	MySQL AB
SQL Server	6.01.7601.17514	Microsoft C

Fig 4.6: Install ODBC Driver

The MySQL ODBC 5.1 Driver was selected and "Finish" was selected. Another user form was filled in with details of the connection to be made.



Connection Parameters

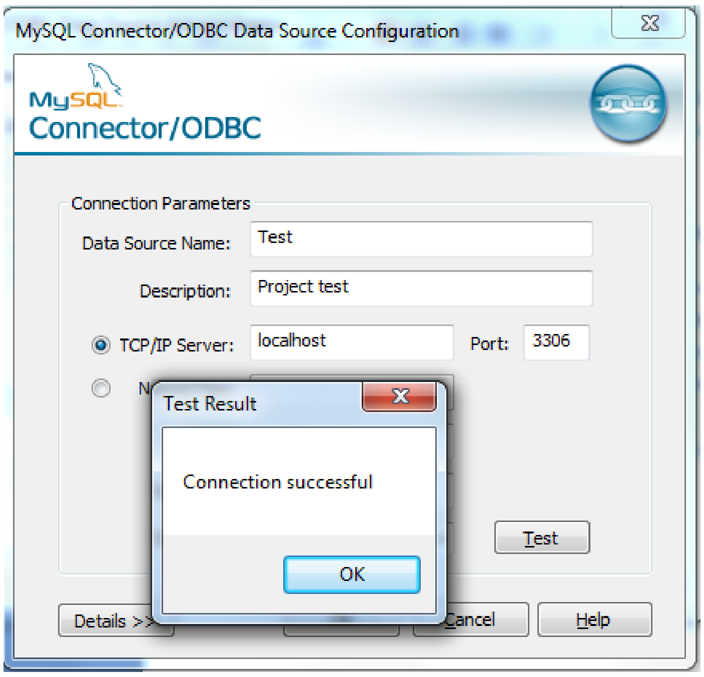
Data Source Name:	Test
Description:	Project test
● TCP/IP Server:	localhost
Port:	3306
○ Named Pipe:	
User:	root
Password:	•••••••••
Database:	project

Fig 4.7: Create ODBC Connection

The form was filled in with the above information:

Data Source Name: The name of the data source
Description: Description of the data source
TCP/IP Server: localhost as the connection is being set up on the local machine
User: Root as this is the user on the local machine
Password: MySQL password.
Database: The database which we wish to connect to in MySQL

We then click "Test" to see if the connection has been successful or not.


MySQL Connector/ODBC Data Source Configuration

Connection Parameters

Data Source Name: Test

Description: Project test

TCP/IP Server: localhost Port: 3306

Test Result

Connection successful

Fig 4.8: ODBC Connection successful

The ODBC DSN has now been set up successfully and connections can be made with qlikview. We will discuss how to connect to qlikview in the next chapter.

5. Qlikview Dashboard

5.1 Introduction

QlikView is a powerful and flexible Business Intelligence platform for turning data into knowledge (Visual Intellegence, 2014). It enables users to easily consolidate, search, and visually analyse all their data for actionable insight. It allows users to:

- Consolidate relevant data from multiple sources into a single application
- Explore the associations in the data
- Enable decision making
- Visualize data with engaging, state-of-the-art graphics
- Search across all data—directly and indirectly

Interact with dynamic apps, dashboards and analytics (Qlik, 2014)

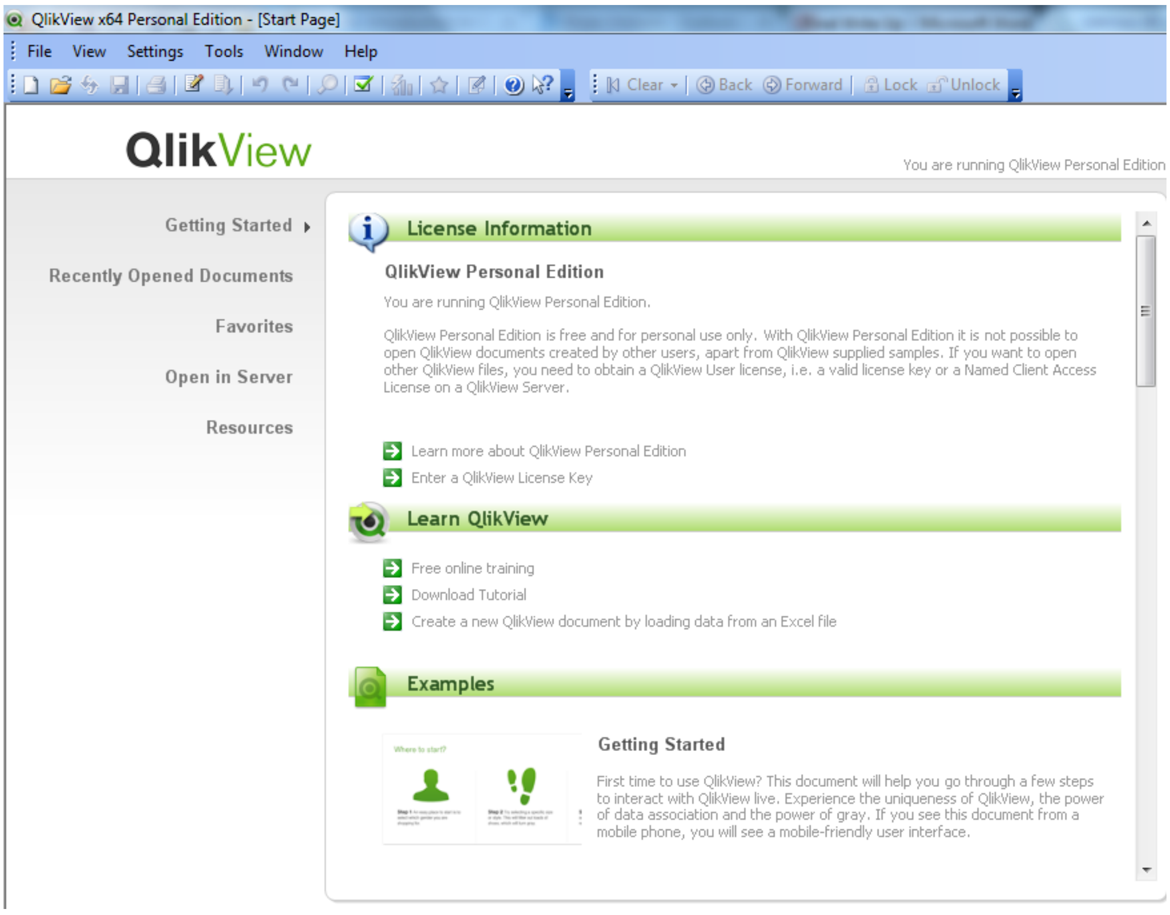
Through the creation of interactive, graphical dashboards qlikview allows the end user to view the data in the way that is most relevant to them cutting out the need for the middle man in the Management Information (MI) and decision processes. It allows users to uncover hidden trends and make discoveries that drive innovative decisions.

5.2 Download

A personal version of qlikview can be downloaded from the website http://www.qlik.com/us/explore/experience/free-download . This allows users to create their own dashboards on their local machines. There is a range of helpful reading material and tutorials on the qlik.com website which prepares the user to create their first simple dashboard. The downside to the personal edition of qlikview is that dashboards created by other users cannot be opened using this edition. A qlikview user license is required for this purpose.

5.3 ODBC Connection

When the application is opened an introduction screen is shown to the user:


QlikView x64 Personal Edition - [Start Page]
File View Settings Tools Window Help
Clear Back Forward Lock Unlock
QlikView
You are running QlikView Personal Edition
Getting Started
Recently Opened Documents
Favorites
Open in Server
Resources
License Information
Learn QlikView
Examples

QlikView Personal Edition

You are running QlikView Personal Edition.

QlikView Personal Edition is free and for personal use only. With QlikView Personal Edition it is not possible to open QlikView documents created by other users, apart from QlikView supplied samples. If you want to open other QlikView files, you need to obtain a QlikView User license, i.e. a valid license key or a Named Client Access License on a QlikView Server.

Learn more about QlikView Personal Edition
Enter a QlikView License Key

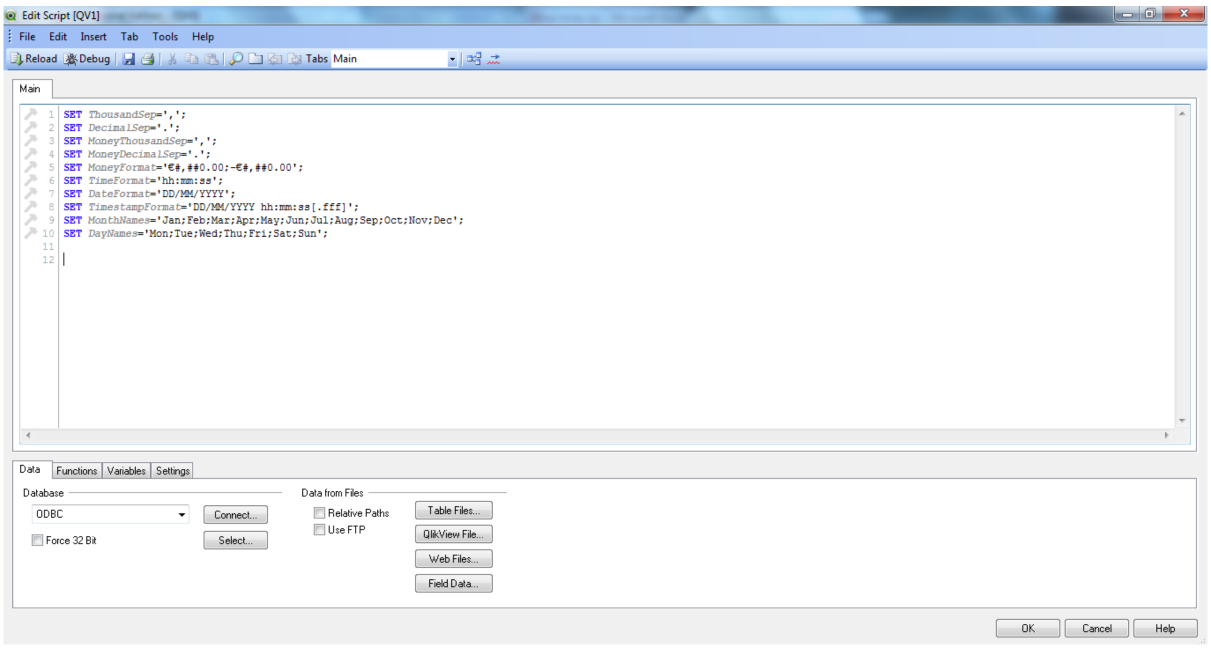
Free online training
Download Tutorial
Create a new QlikView document by loading data from an Excel file

Getting Started

First time to use QlikView? This document will help you go through a few steps to interact with QlikView live. Experience the uniqueness of QlikView, the power of data association and the power of gray. If you see this document from a mobile phone, you will see a mobile-friendly user interface.

Fig 5.1: Qlikview Introduction dashboard

To create a new qlikview document the “New” tab is opened and a blank canvas appears. This is where all of the graphs and tables etc will be created from the data. Before these are created the data must first be loaded in. The “Edit Script” tab is selected and the following screen appears to the user:



```
SET ThousandSep=',';
SET DecimalSep='.';
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='€#,##0.00;-€#,##0.00';
SET TimeFormat='hh:mm:ss';
SET DateFormat='DD/MM/YYYY';
SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff]';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';
```

Fig 5.2: Qlikview Load Editor

In order to create the ODBC connection with the DSN created in the previous section, ODBC is selected from the drop down box in the Data tab and the click Connect. The following options are presented:

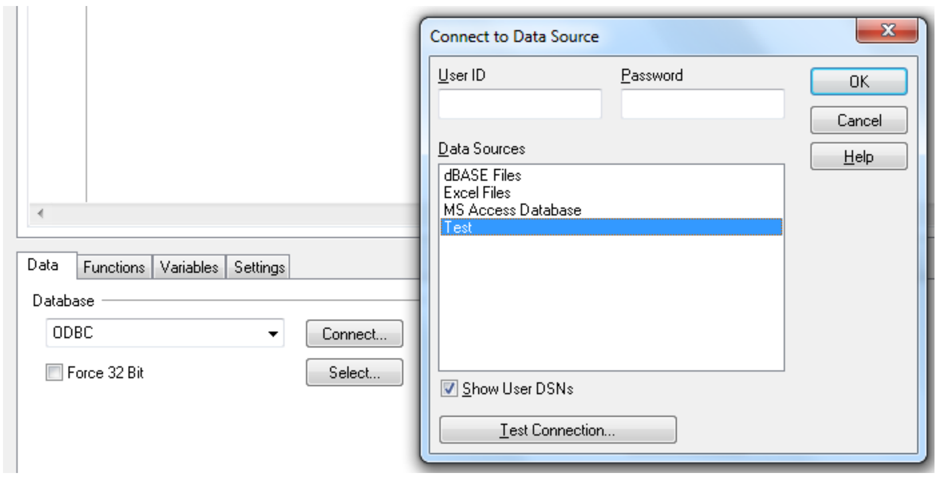


Fig 5.3: ODBC Connection

The DSN which was created was called Test and this is selected from the options. When this is done the document should be saved to a location on the local machine. Once this is complete the connection has been made and the data from the MySQL database can be loaded into the application.

5.4 Data Load

The data was loaded in using the script editor in Qlikview. The syntax for loading in the tables is very similar to SQL. An example of the loading of one of the tables is shown below:

```
PLAYER:
LOAD
GAME_DTE,
GAMEWEEK,
LOOKUP('TEAM_NAME', 'TEAM_ABBRV', OPPOSITION, 'TEAM') AS OPPOSITION_NAME,
LOOKUP('TEAM_POSITION', 'TEAM_NAME', LOOKUP('TEAM_NAME', 'TEAM_ABBRV', OPPOSITION,
'TEAM'), 'LEAGUE') AS OPPOSITION_POSITION,
HOME_AWAY,
SCORE as RESULT,
MATCH_OUTCOME,
MINS_PLYD,
GOALS_SCORED,
ASSISTS,
CLEAN_SHEET,
GOALS_CONCEDED,
OWN_GOALS,
PENALTIES_SAVED,
PENALTIES_MISSED,
YELLOW_CARDS,
RED_CARDS,
SAVES,
BONUS,
EA_PPI,
BONUS_POINTS_SYS,
NET_TRANSFERS,
PLAYER_VALUE/ 10 as PLAYER_VALUE, //Value appears as a multiple of 10 in the file
POINTS,
PLAYER_NAME,
TEAM_NAME,
POSITION,
PRICE/10 as PRICE,
SELECTED_BY,
PLAYER_ID
;
SQL SELECT *
FROM PLAYER_HISTORY
;
```

In this case the table PLAYER is loaded in and is taken from the SQL table PLAYER_HISTORY. Some changes have been made to the table fields in PLAYER_HISTORY. The lookup function will be explained at a later stage. The script for loading in all of the data is shown in the Appendix G.

The script is executed by clicking on the “Reload” button.

5.4.1 Schema, Joins and Associations

When the data load is complete the schema of the tables can be viewed using CTRL + t. The schema is shown below:

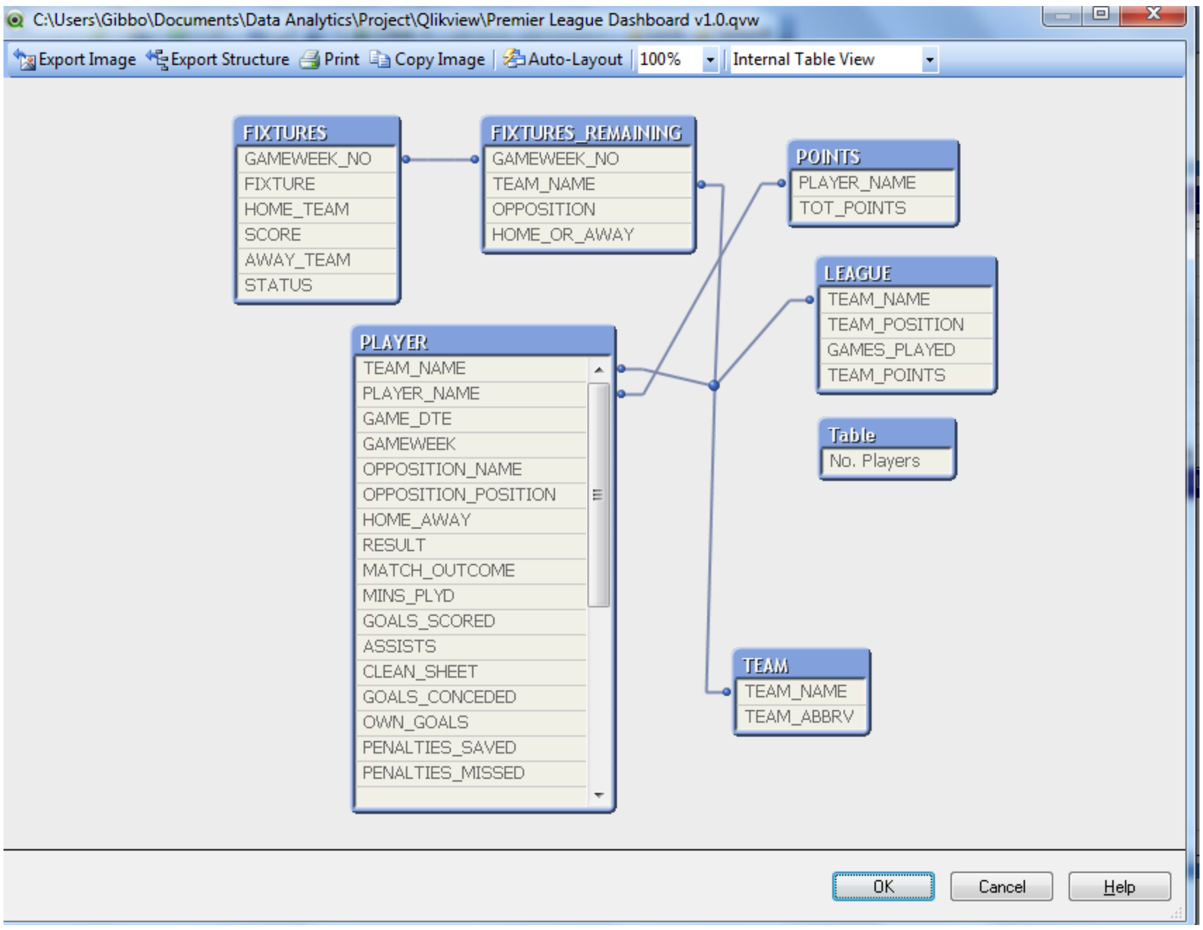


Fig 5.4: Qlikview Databse Schema

Unlike SQL where a join is explicitly defined between tables and columns, in Qlikview an associative join occurs between tables where they share fields with the same names. The QlikView internal logic allows a multi-table data model. Tables are linked by the naming of the keys so that QlikView "knows" how to evaluate the relations at run-time. The associations are evaluated as joins at the moment when the user clicks in the application, making a selection (Qlik, 2012). That means that these links are implicit joins that have not yet been made.

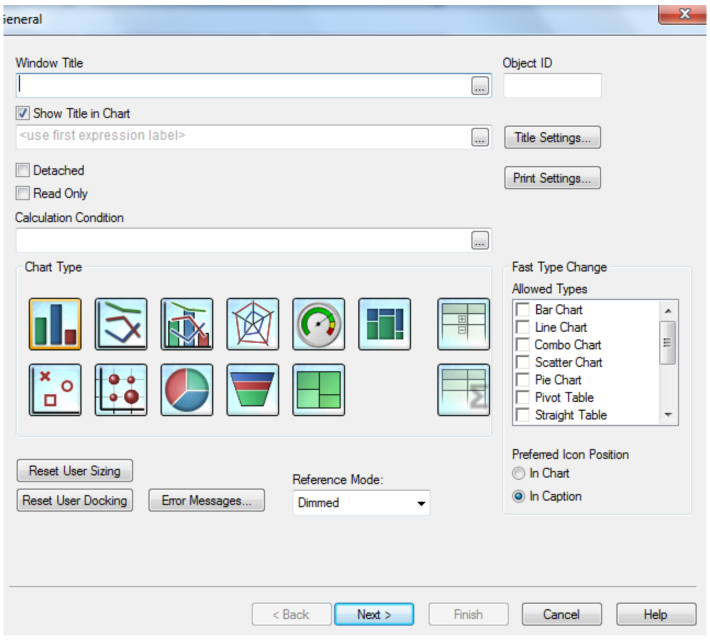
The Qlikview script can also contain actual joins where two or more tables are joined in the load script and the end result is a single table which is a function of the join condition and selections from the tables. This is not evaluated on selection by the user.

Hence, the main difference between the associations and joins is that the associations are evaluated at demand; as the user makes selections. As opposed to the joins that are evaluated when the script runs.

5.5 Creating Charts and Tables

There are a number of different types of charts and tables available in qlikview. The data is generally split up into dimensions and expressions. The dimensions are generally the range over which you wish to make the calculation on the data. The expression data is the metric on which a certain calculation is performed. For example, if a business user wanted to find the volume of sales per month, the dimension would be the month value and the expression would be the sum of the sales. The general rule is that you will have a data point for every dimensional value in a chart so in this case there would be a value for the sum of the sales for each of the months in the year.

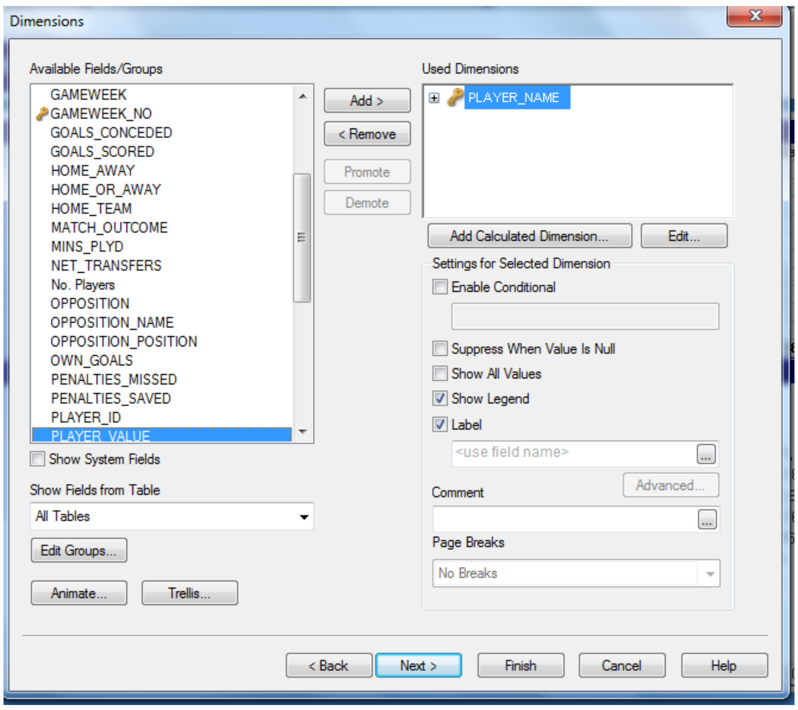
In order to create a chart, the dimensions and expressions must be defined. The user must right click on the canvas and select “New Sheet Object”. The type of chart/table can then be selected from the drop down menu.



[✓] Show Title in Chart
<use first expression label>
Reference Mode: Dimmed
Preferred Icon Position: [✓ In Caption]

Fig 5.5: Qlikview Graph Type Selection

The entire range of fields that have been loaded into the qlikview session are available for selection as dimensions.



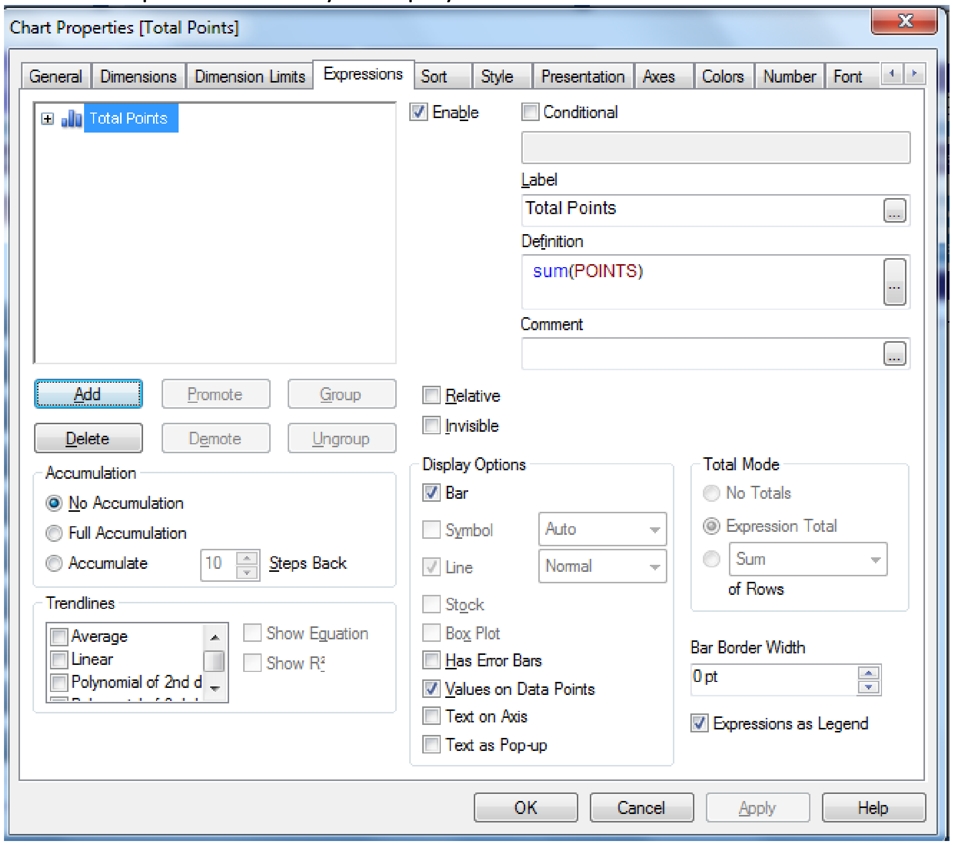
Used Dimensions: PLAYER_NAME

☑ Show Legend

☑ Label

Fig 5.6:Dimension Selection

In this instance the dimension selected is the field PLAYER_NAME. The expression is then taken as the overall sum of the points scored by each player.



Expression: Total Points

☑ Enable

Label: Total Points

Definition: sum(POINTS)

Accumulation: ◉ No Accumulation

Accumulate: 10 Steps Back

Display Options: ☑ Bar, ☑ Values on Data Points

Total Mode: ◉ Expression Total

Bar Border Width: 0 pt

☑ Expressions as Legend

Fig 5.7: Expression Selection

This provides us with a bar graph that shows every player's total points in the game.

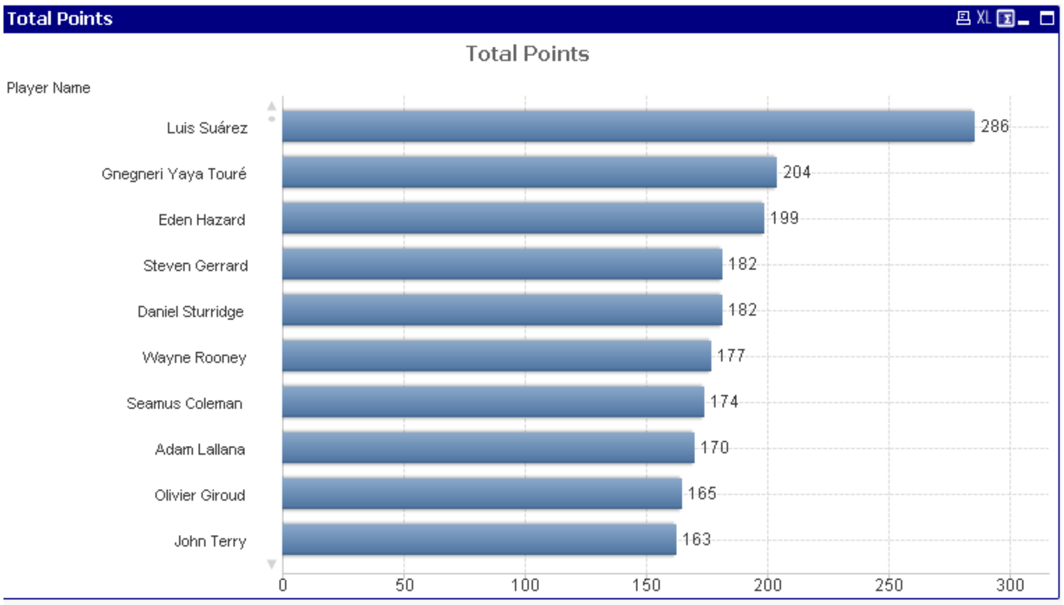
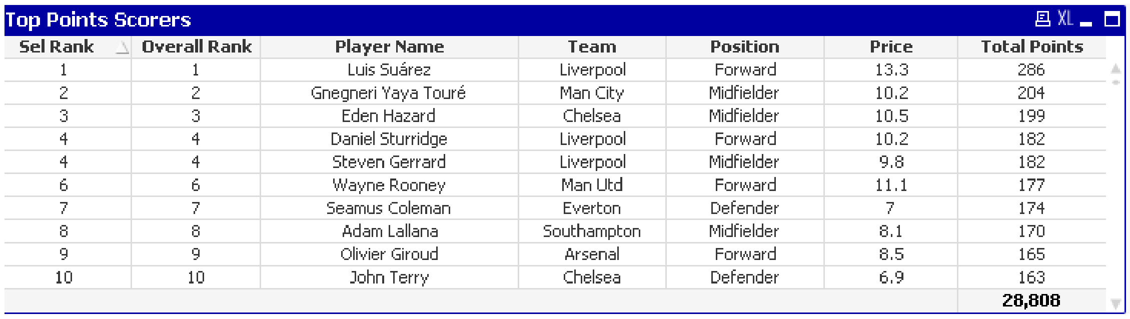


Fig 5.8: Total Points Scored

The number of dimensions shown on the screen can be limited by using the dimension limits tab in the graph creation screen. This can then allow the user to scroll down through all of the players with each name and value clearly legible.

A table can be created in the same way with dimensions and expressions. This type of table is called a straight table. A table box is different in that there are no expressions and the user just selects the fields they wish to display in the table. The below table is a straight table as it has the calculated dimensions called selection rank and overall rank.



Top Points Scorers

Sel Rank	Overall Rank	Player Name	Team	Position	Price	Total Points
1	1	Luis Suárez	Liverpool	Forward	13.3	286
2	2	Gnegneri Yaya Touré	Man City	Midfielder	10.2	204
3	3	Eden Hazard	Chelsea	Midfielder	10.5	199
4	4	Daniel Sturridge	Liverpool	Forward	10.2	182
4	4	Steven Gerrard	Liverpool	Midfielder	9.8	182
6	6	Wayne Rooney	Man Utd	Forward	11.1	177
7	7	Seamus Coleman	Everton	Defender	7	174
8	8	Adam Lallana	Southampton	Midfielder	8.1	170
9	9	Olivier Giroud	Arsenal	Forward	8.5	165
10	10	John Terry	Chelsea	Defender	6.9	163
						28,808

Fig 5.9: Master Points Table

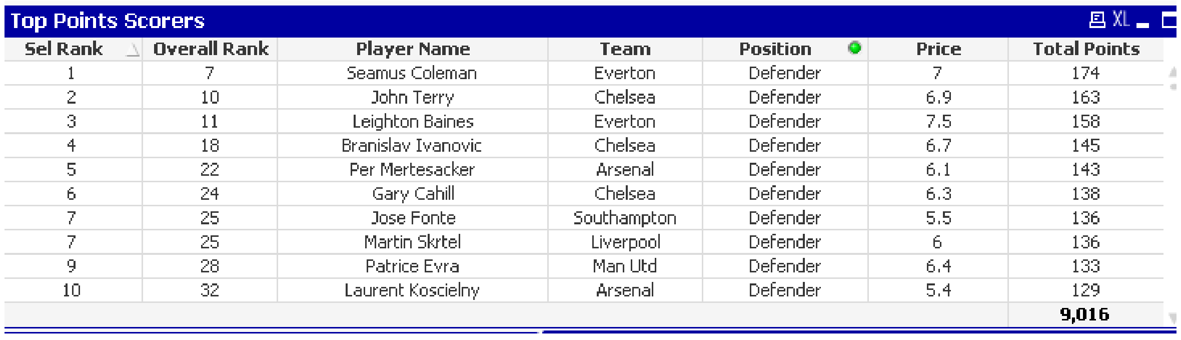
The calculated dimension for the Selection rank is:

=Aggr(Rank(Sum(POINTS),1,1),PLAYER_NAME)

And the calculated dimension for the Overall rank is

=Aggr({1} Rank(Sum({1}POINTS),1,1),PLAYER_NAME)

The difference between the two of these is that when a user makes a selection such as team name or position, the overall rank will stay the same while the selection rank will update the calculation based on the selection that has been made. The {1} in the code for the second calculated dimension tells it to ignore all selections made. So for example, if only defenders were selected the table would look like the following:


Top Points Scorers

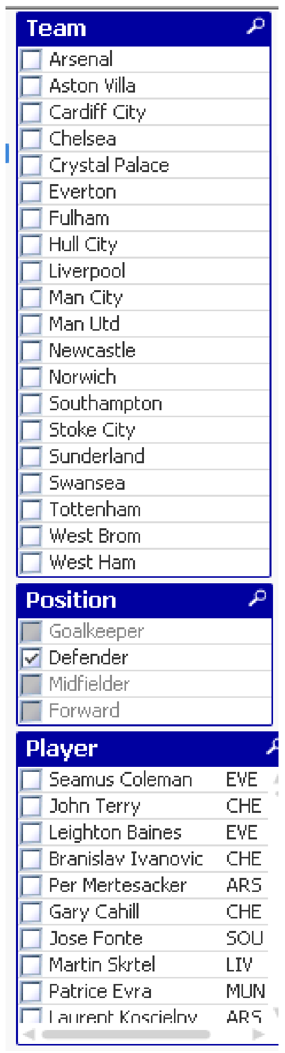
Sel Rank	Overall Rank	Player Name	Team	Position	Price	Total Points
1	7	Seamus Coleman	Everton	Defender	7	174
2	10	John Terry	Chelsea	Defender	6.9	163
3	11	Leighton Baines	Everton	Defender	7.5	158
4	18	Branislav Ivanovic	Chelsea	Defender	6.7	145
5	22	Per Mertesacker	Arsenal	Defender	6.1	143
6	24	Gary Cahill	Chelsea	Defender	6.3	138
7	25	Jose Fonte	Southampton	Defender	5.5	136
7	25	Martin Skrtel	Liverpool	Defender	6	136
9	28	Patrice Evra	Man Utd	Defender	6.4	133
10	32	Laurent Koscielny	Arsenal	Defender	5.4	129
						9,016

Fig 5.10: Master Points Table with selections

From this we can see that Seamus Coleman is the top points scoring defender and is 7th in the overall ranking. Also at the bottom of the table we can see that the total number of points scored by all defenders is 9,016. This is an option when creating the table to have totals or not.

5.6 Making Selections

As discussed previously, an associative join occurs between tables where they share fields with the same names. When certain selections are made, this implicit join is evaluated and the results are displayed to the screen. In order to make the process of making selections easier, list boxes were used in the dashboard which gives the user the option to selected whatever data they require. Three main list boxes were used for every tab in the dashboard – Team Name, Position and Player Name



Team

- Arsenal
- Aston Villa
- Cardiff City
- Chelsea
- Crystal Palace
- Everton
- Fulham
- Hull City
- Liverpool
- Man City
- Man Utd
- Newcastle
- Norwich
- Southampton
- Stoke City
- Sunderland
- Swansea
- Tottenham
- West Brom
- West Ham

Position

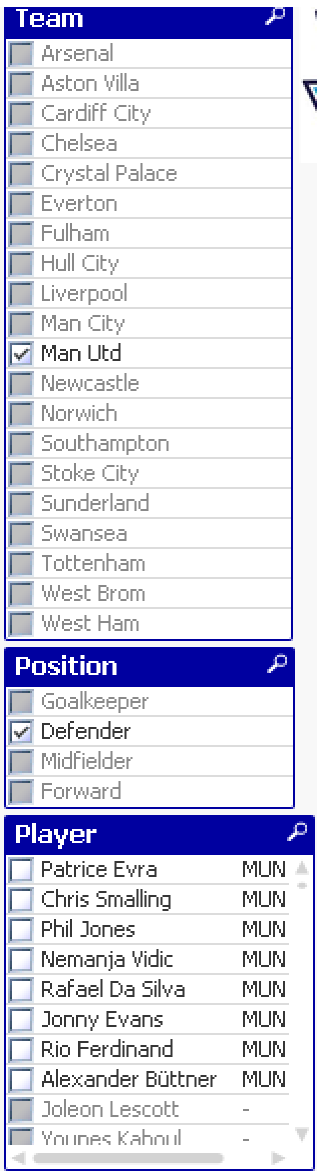
- Goalkeeper
- ✓ Defender
- Midfielder
- Forward

Player

Seamus Coleman	EVE
John Terry	CHE
Leighton Baines	EVE
Branislav Ivanovic	CHE
Per Mertesacker	ARS
Gary Cahill	CHE
Jose Fonte	SOU
Martin Skrtel	LIV
Patrice Evra	MUN
Laurent Koscielny	ARS

Fig 5.11: Selection Boxes

In the example above, the position Defender is selected. This means that only defenders appear in the Player name list box. If the user wished to only display defenders for Manchester United then the following would be displayed:



Team
- [ ] Arsenal
- [ ] Aston Villa
- [ ] Cardiff City
- [ ] Chelsea
- [ ] Crystal Palace
- [ ] Everton
- [ ] Fulham
- [ ] Hull City
- [ ] Liverpool
- [ ] Man City
- [✓] Man Utd
- [ ] Newcastle
- [ ] Norwich
- [ ] Southampton
- [ ] Stoke City
- [ ] Sunderland
- [ ] Swansea
- [ ] Tottenham
- [ ] West Brom
- [ ] West Ham

Position
- [ ] Goalkeeper
- [✓] Defender
- [ ] Midfielder
- [ ] Forward

Player

Patrice Evra	MUN
Chris Smalling	MUN
Phil Jones	MUN
Nemanja Vidic	MUN
Rafael Da Silva	MUN
Jonny Evans	MUN
Rio Ferdinand	MUN
Alexander Büttner	MUN
Joleon Lescott	-
Younes Kaboul	-

Fig 5.12: Selection Boxes with Selections

The graphs in the dashboard will be updated accordingly also. For example, the graph which was created to show the goals scored by each player would now look like this.

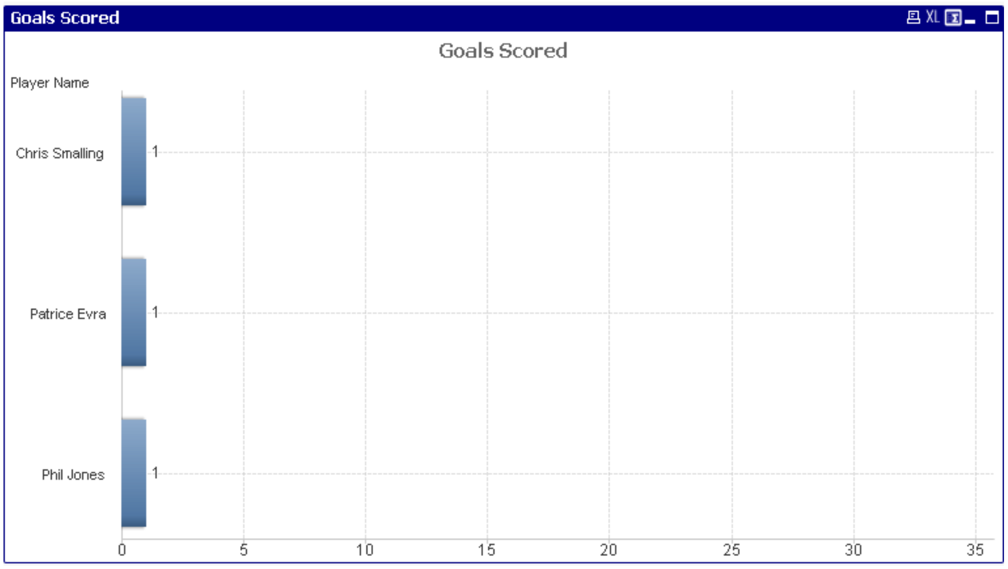
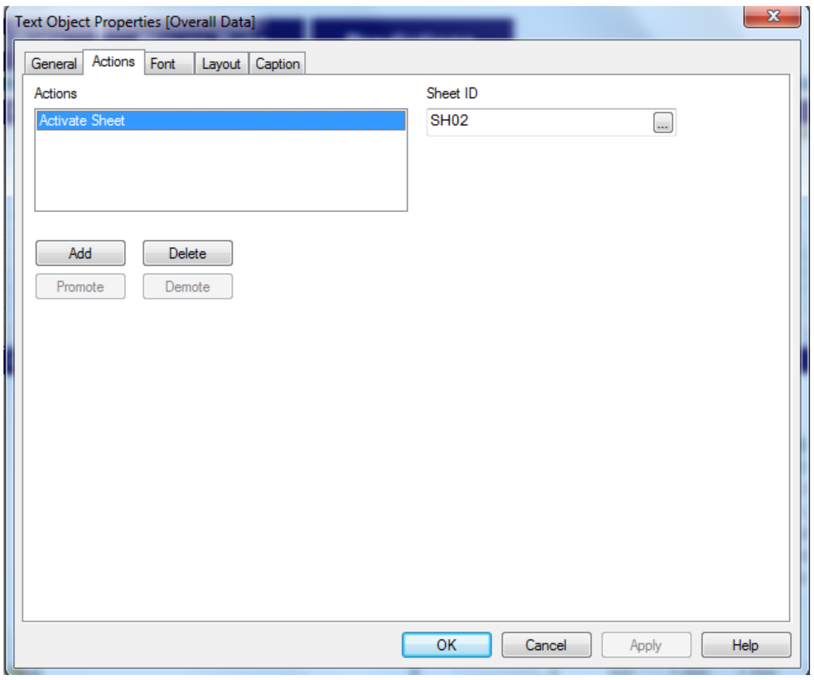


Fig 5.13: Graph with selection

From this we can see that only three Manchester United defenders have scored goals. These selections can be made to filter the data as the user wishes.

5.7 Switching Between Sheets

In order to create more real estate for the qlikview objects, a number of different sheets were created which allowed for the segregation of the graphs and tables into different sections. When a sheet is created an action can be created on an object so that when it is selected a certain task is carried out. In order to switch between sheets a number of text objects were created and the action on these was to activate a certain sheet. This was done in the Actions tab of the object:

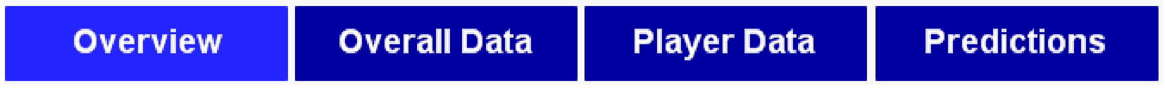

Text Object Properties [Overall Data]

Actions: Activate Sheet

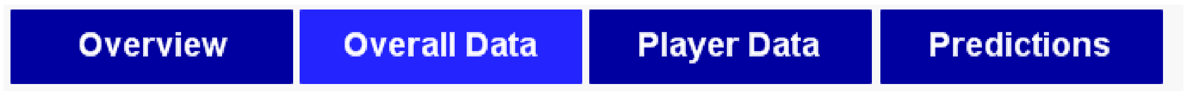
Sheet ID: SH02

Fig 5.14: Set Actions

The sheet ID of the required sheet was set and when the text object was clicked the set sheet was activated. In the dashboard, 4 main tabs were created for the 4 sheets. These can be seen below:



As each tab is selected, the colour of the text object changes to highlisght which sheet the user is currently using. This is done using conditional formatting in qlikview.



When the tab is selected the graphs on that particular sheet will appear.

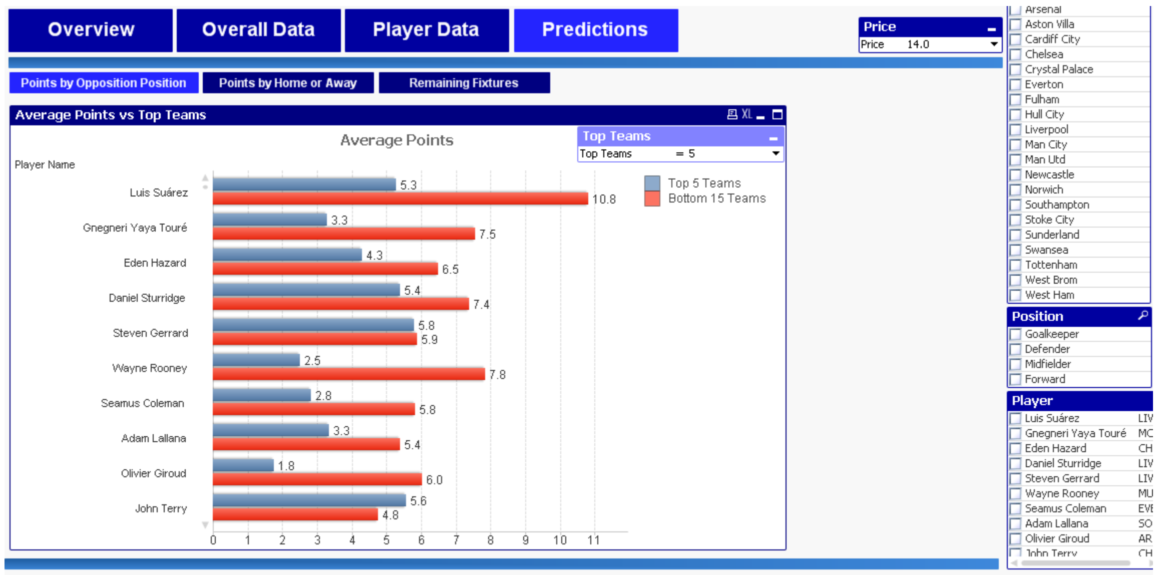
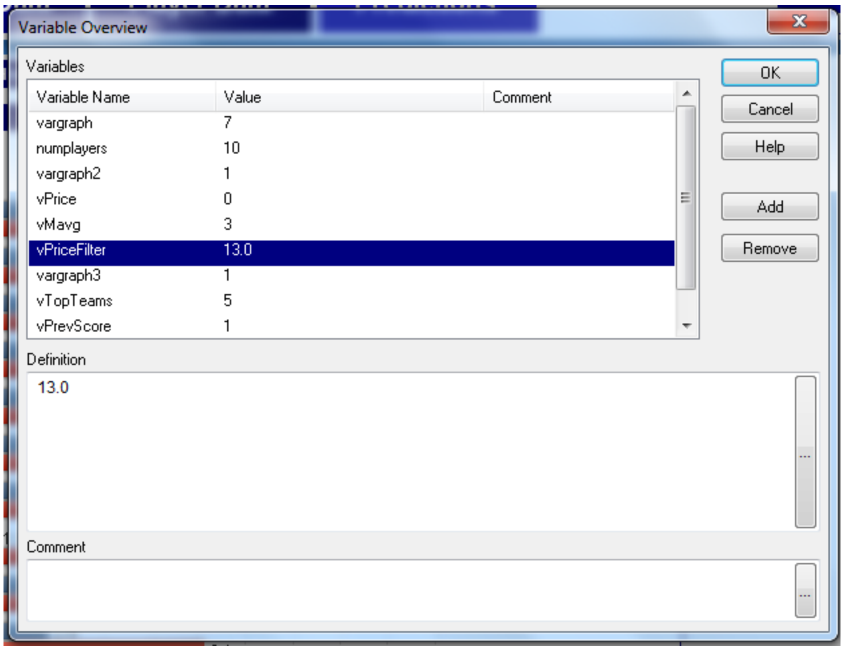


Fig 5.15: Tab selection

5.8 Setting Variables

Variables are a very useful feature in qlikview. They allow the developer to create a set or changeable value based on a formula or just a hard coded value. These variables can then be referenced in the creation of the qlikview objects to limit certain features or include in the calculation of an expression.

To create a variable the Variable Overview box was selected from settings.


Variable Overview

Variables

Variable Name	Value	Comment
vargraph	7	
numplayers	10	
vargraph2	1	
vPrice	0	
vMavg	3	
vPriceFilter	13.0	
vargraph3	1	
vTopTeams	5	
vPrevScore	1	

Definition

13.0

Comment

Fig 5.15: Set variables

The variable vPriceFilter was created to place a limit on the players returned in the graphs and the selection boxes based on the players' respective values. This allows the user to put a range on the price of the players that are returned if they have a limited transfer budget.

The price filter was implemented in the form of a drop down input box where the value selected set the vPriceFilter to that value.

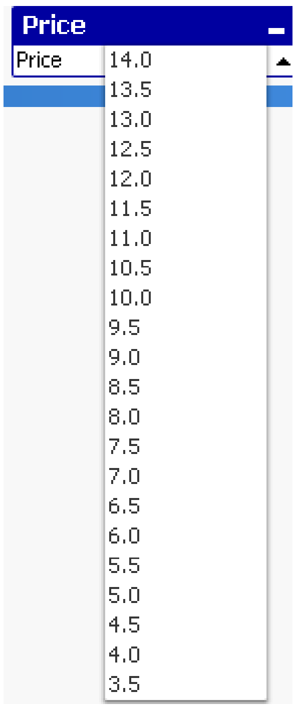


Fig 5.16: Set Variable value

When a value is selected from this dropdown the players returned in the graphs and selection boxes are only those who have a value equal to or less that the value selected in the drop down. When the value is set to the maximum all players are returned.

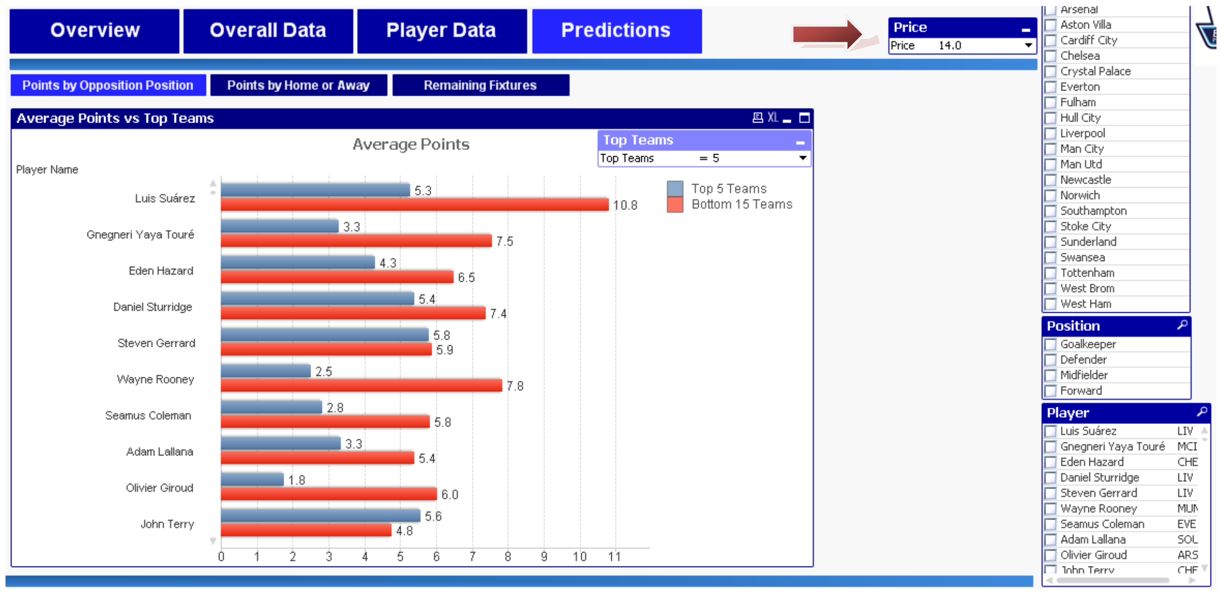


Fig 5.17: Tab selection with max price

When the value is set to a lower price only players with that price or lower are returned.

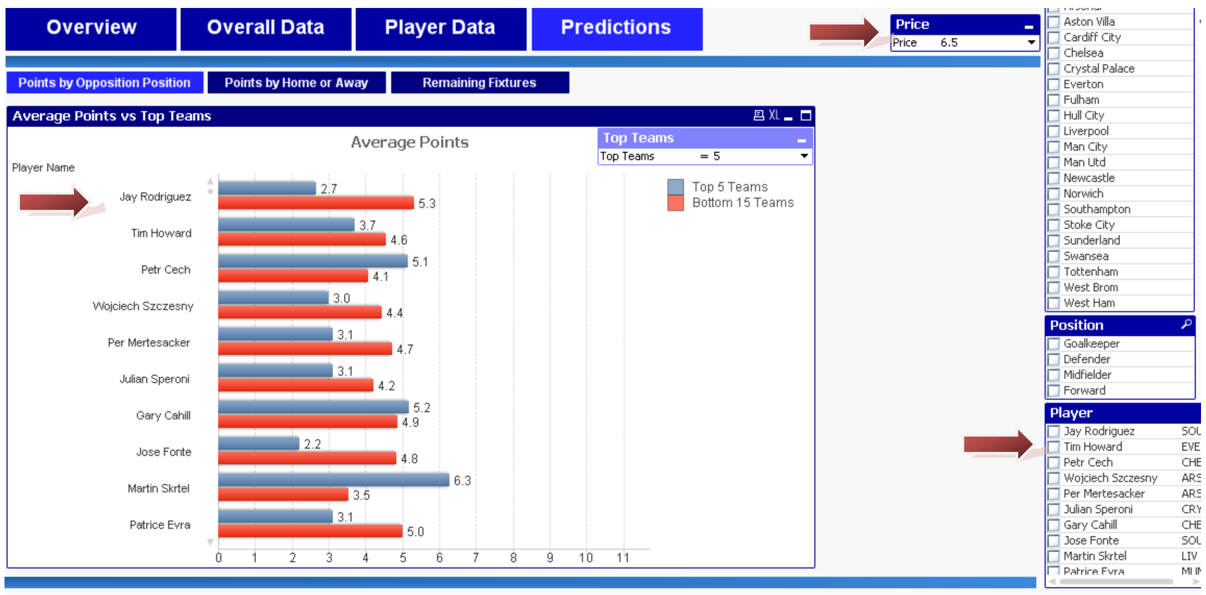
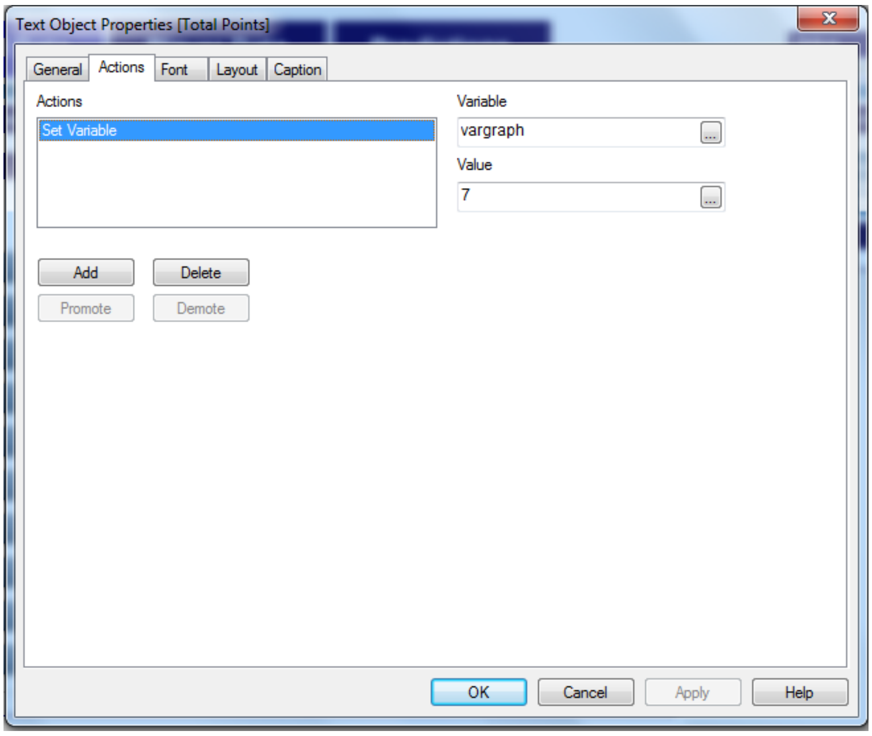


Fig 5.18: Tab selection with price filter

Variables were also used to show certain graphs when certain conditions are met. The sub tabs in the sheets were created as sheet objects and an action created on each object to set a certain variable to a certain value.



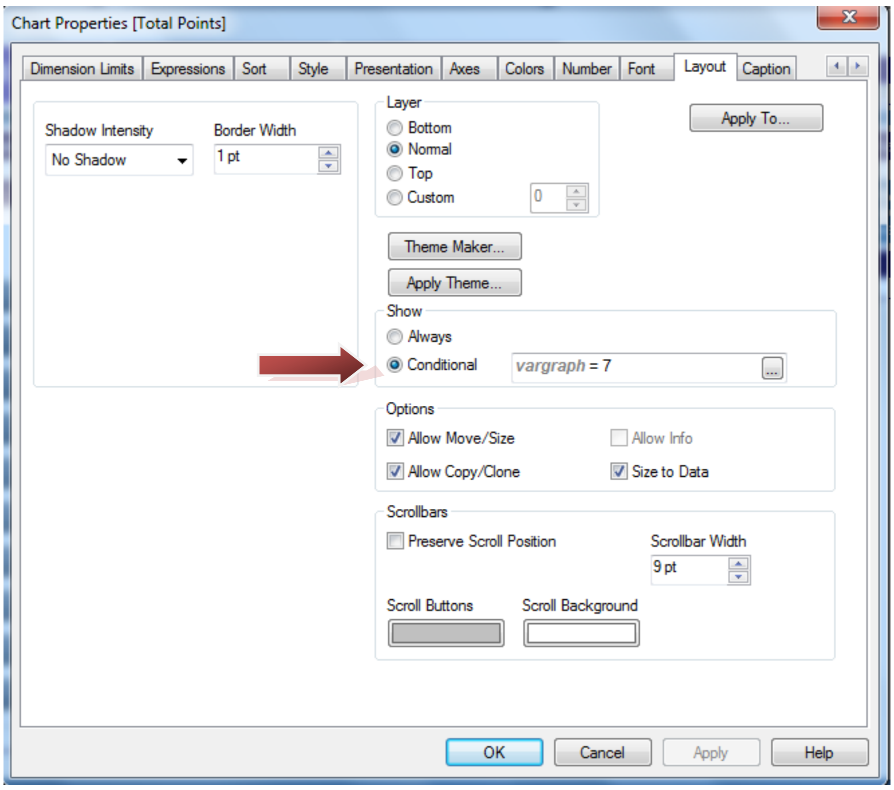
Actions: Set Variable

Variable: vargraph

Value: 7

Fig 5.19: Set variable to text objects

In this instance by selecting the object the variable vargraph was set to a value of 7. In a graph in the sheet a rule was applied to only show that graph when the value of the vargraph variable = 7.



Shadow Intensity: No Shadow

Border Width: 1 pt

Layer: Bottom [✓ Normal] Top Custom 0

Show: Always [✓ Conditional] vargraph = 7

Options: [✓ Allow Move/Size] Allow Info [✓ Allow Copy/Clone] [✓ Size to Data]

Scrollbars: Preserve Scroll Position

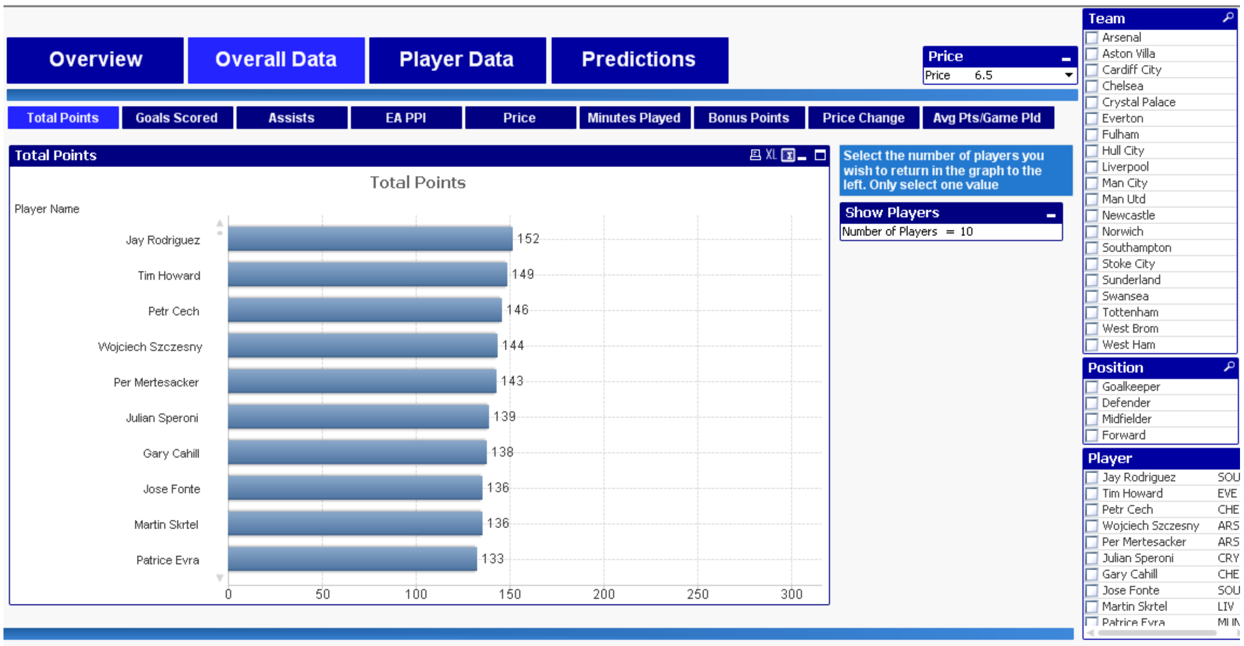
Scrollbar Width: 9 pt

Scroll Buttons

Scroll Background

Fig 5.20: Conditional Layout

When that tab is selected, only that graph will appear.



Price 6.5

Select the number of players you wish to return in the graph to the left. Only select one value

Number of Players = 10

Fig 5.21: Select tabs

This applies for all sub tabs with each one of these highlighting a single graph to the page.

5.9 Key Performance Indicators

At the beginning of the dashboard development a number of potential users were interviewed as to what features and KPIs they would like to see included in the dashboard. The majority of these were the same ones available in the website itself but with an emphasis on a nicer more user friendly graphical interface which would allow the user to navigate their way through the system and easily identify targets for transfers and substitutions.

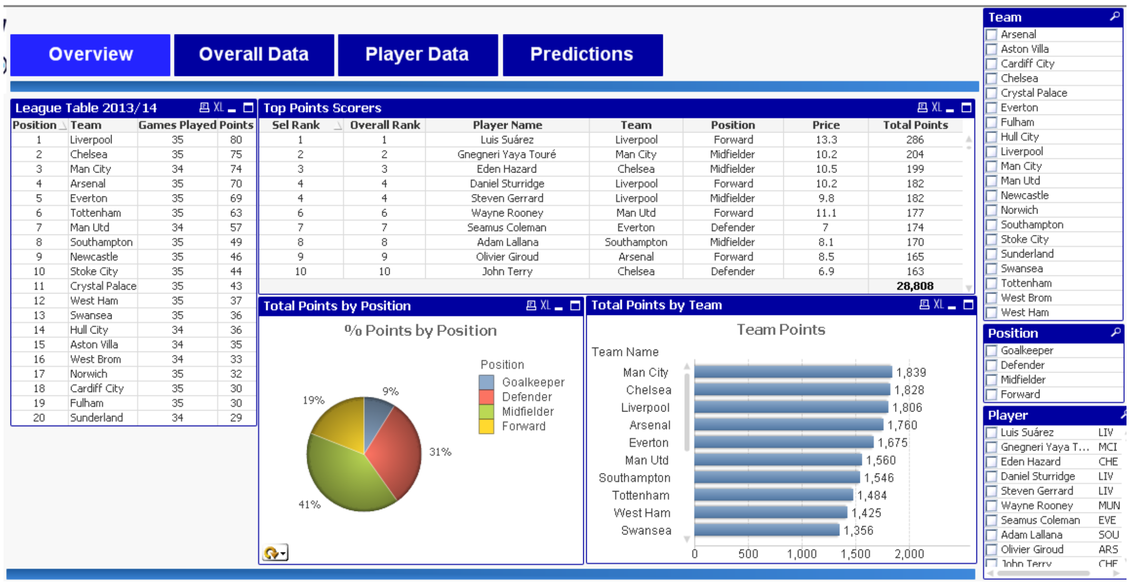
Other aspects identified included:

- Functionality to view a moving average of players scoring history which allowed the user to get a clearer picture of the players form.
- An ability to compare players scoring across game weeks with other players in the game
- Functionality to see a players average scoring in home games vs. away games
- Functionality to see a player's average scoring against the top teams vs. the bottom teams.

5.9.1 Dashboard Tabs

5.9.1.1 Overview

Each of the tabs on the dashboard attempts to provide the required functionality to the user. The first tab is the Overview sheet which gives a high level overview of the statistics in the game. It contains the up to date league table, a table with all of the players sorted by highest scoring to lowest scoring, a pie chart with the percentage of the total points scored by position and a bar graph showing the top scoring teams in the game.



League Table 2013/14

Position	Team	Games Played	Points
1	Liverpool	35	80
2	Chelsea	35	75
3	Man City	34	74
4	Arsenal	35	70
5	Everton	35	69
6	Tottenham	35	63
7	Man Utd	34	57
8	Southampton	35	49
9	Newcastle	35	46
10	Stoke City	35	44
11	Crystal Palace	35	43
12	West Ham	35	37
13	Swansea	35	36
14	Hull City	34	36
15	Aston Villa	34	35
16	West Brom	34	33
17	Norwich	35	32
18	Cardiff City	35	30
19	Fulham	35	30
20	Sunderland	34	29

Top Points Scorers

Sel Rank	Overall Rank	Player Name	Team	Position	Price	Total Points
1	1	Luis Suárez	Liverpool	Forward	13.3	286
2	2	Gnegneri Yaya Touré	Man City	Midfielder	10.2	204
3	3	Eden Hazard	Chelsea	Midfielder	10.5	199
4	4	Daniel Sturridge	Liverpool	Forward	10.2	182
4	4	Steven Gerrard	Liverpool	Midfielder	9.8	182
6	6	Wayne Rooney	Man Utd	Forward	11.1	177
7	7	Seamus Coleman	Everton	Defender	7	174
8	8	Adam Lallana	Southampton	Midfielder	8.1	170
9	9	Olivier Giroud	Arsenal	Forward	8.5	165
10	10	John Terry	Chelsea	Defender	6.9	163
						28,808

Fig 5.22: Overview Tab

5.9.1.2 Overall Data

The overall data tab contains information about the top scoring players across a range of metrics. These metrics are:

- Total Points
- Goals Scored
- Assists
- EA Player Performance Index (PPI)
- Price
- Minutes Played
- Bonus Points
- Price Change
- Average Points per Game

Each sub tab in this sheet relates to a different graph. The image below shows the graph relating to the Price Change metric. The players are split into the top price risers and the lowest price fallers. Two buttons are added to the graph which allows the user to view either one or the other.

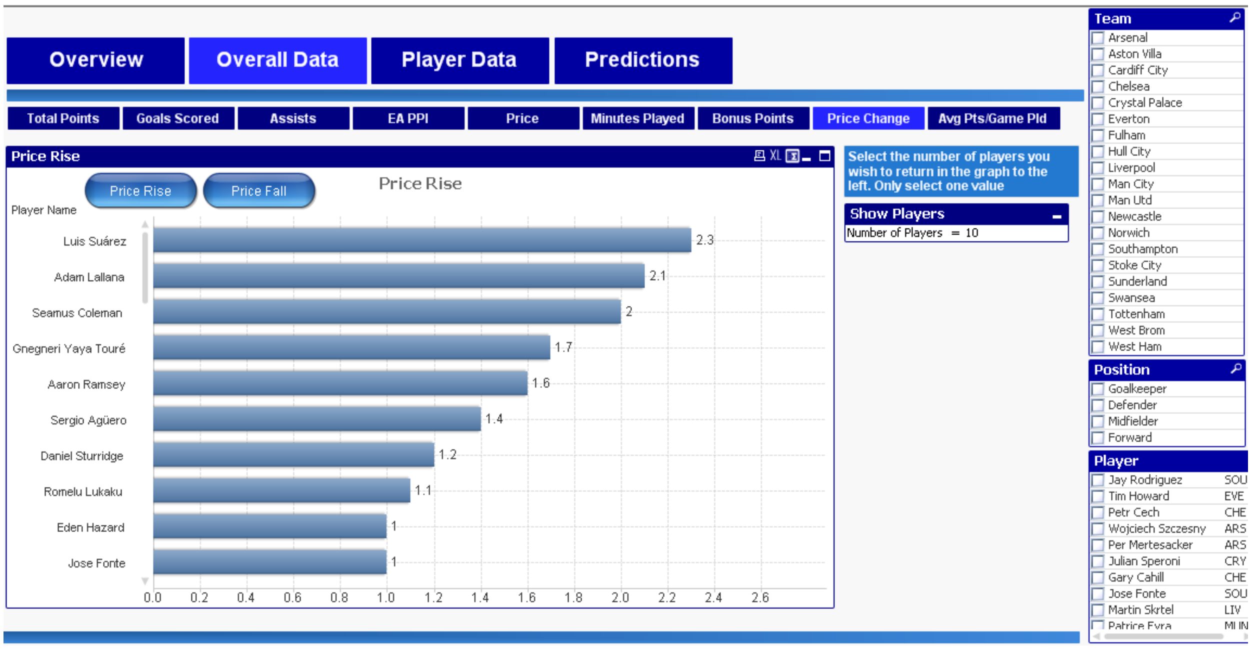


Fig 5.23: Overall Data – Price Rise Graph

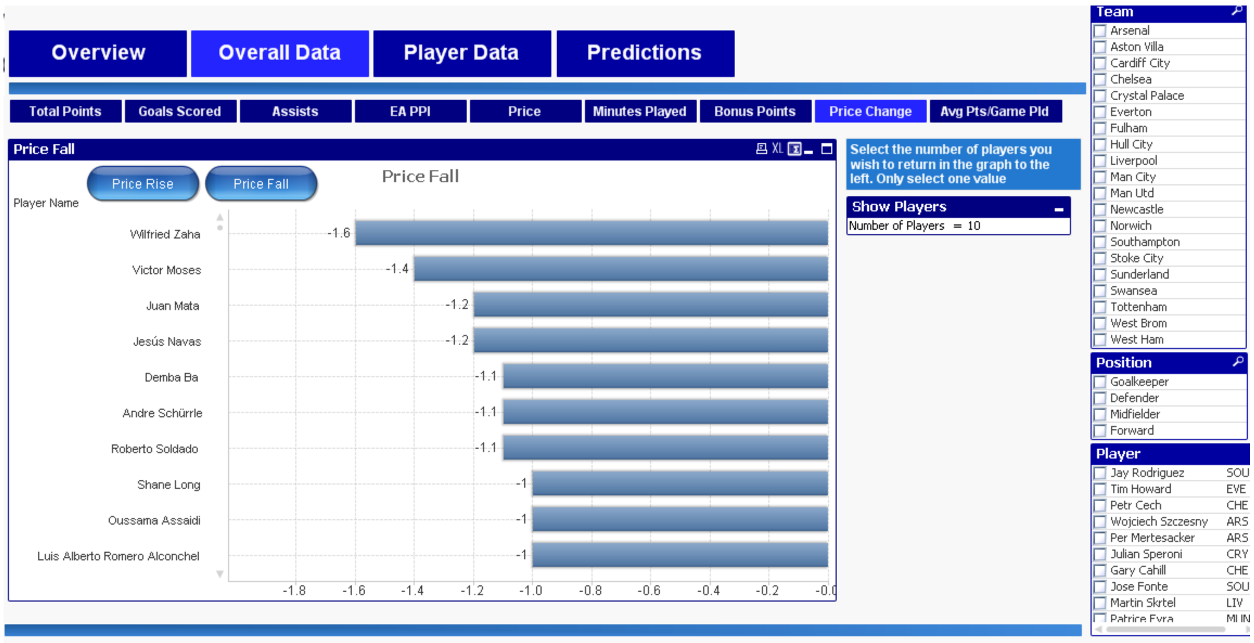


Fig 5.24: Overall Data – Price Fall Graph

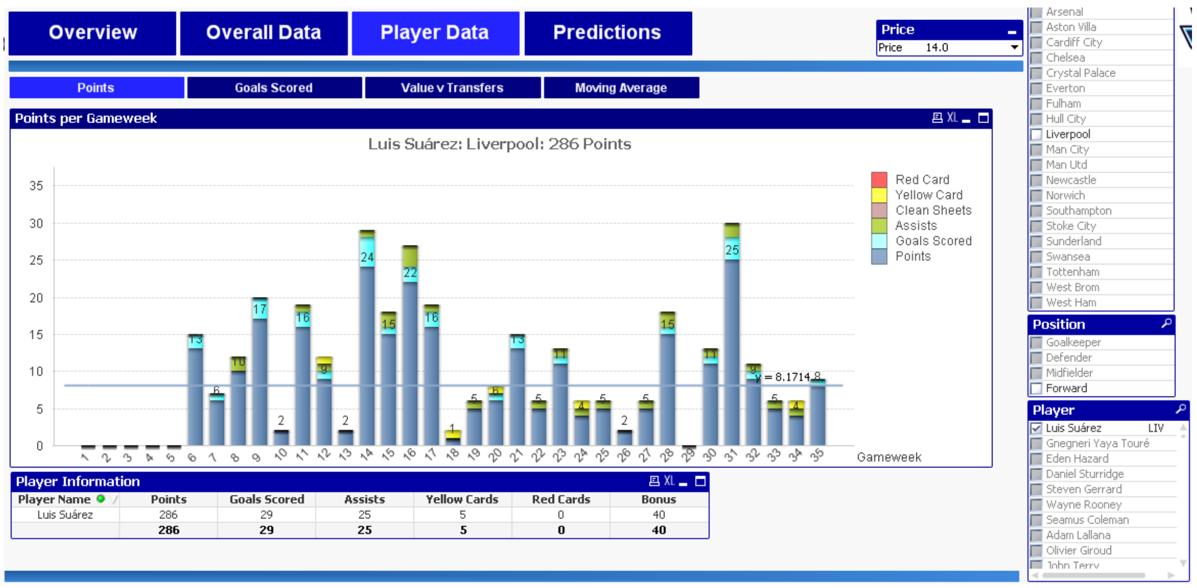
5.9.1.3 Player Data

The Player Data tab allows the user to view certain information about whichever players they choose. In order to show a graph, a player must be selected from the selection box along the side. The metrics in this section are split up into 4:

- Points
- Goals Scored
- Value v Transfers

- Moving Average

The data in this section shows a time series data on a game week level. The dist sub tab in the section shows the players scoring history in a graphical manner showing the points scored, goals scored, yellow/red cards received assists and clean sheets if the player is a defender or goalkeeper.

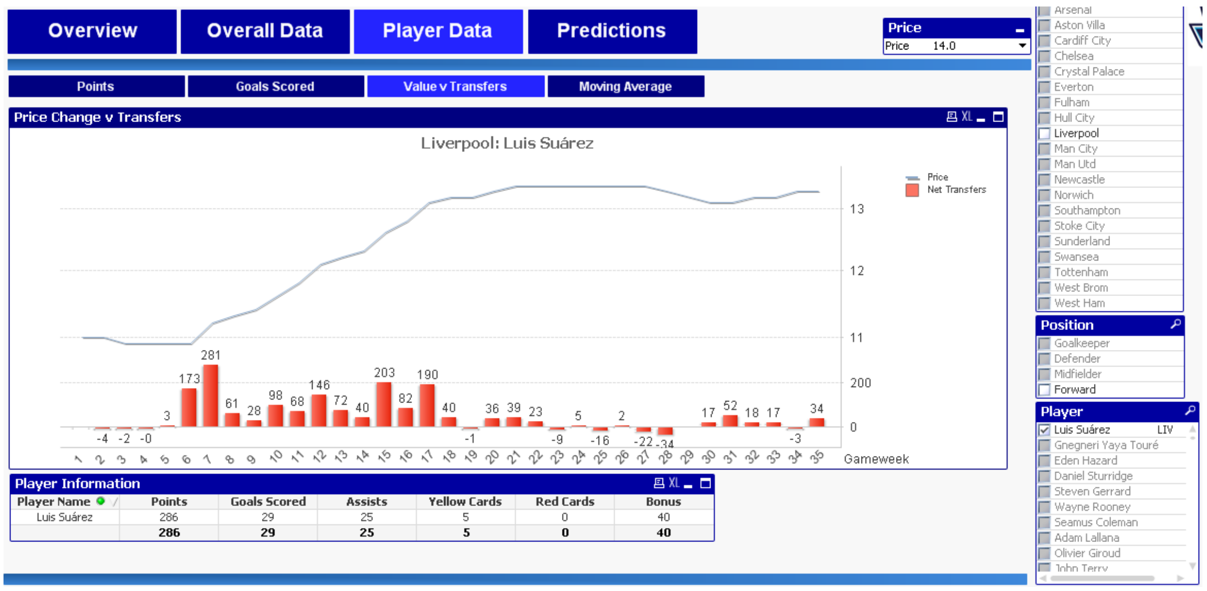


Player Information

Player Name	Points	Goals Scored	Assists	Yellow Cards	Red Cards	Bonus
Luis Suárez	286	29	25	5	0	40
	286	29	25	5	0	40

Fig 5.25: Player Data – Points per Game week

The Value v Transfers sub-tab shows how the player's value relates and responds to the net transfers of that player in the game.

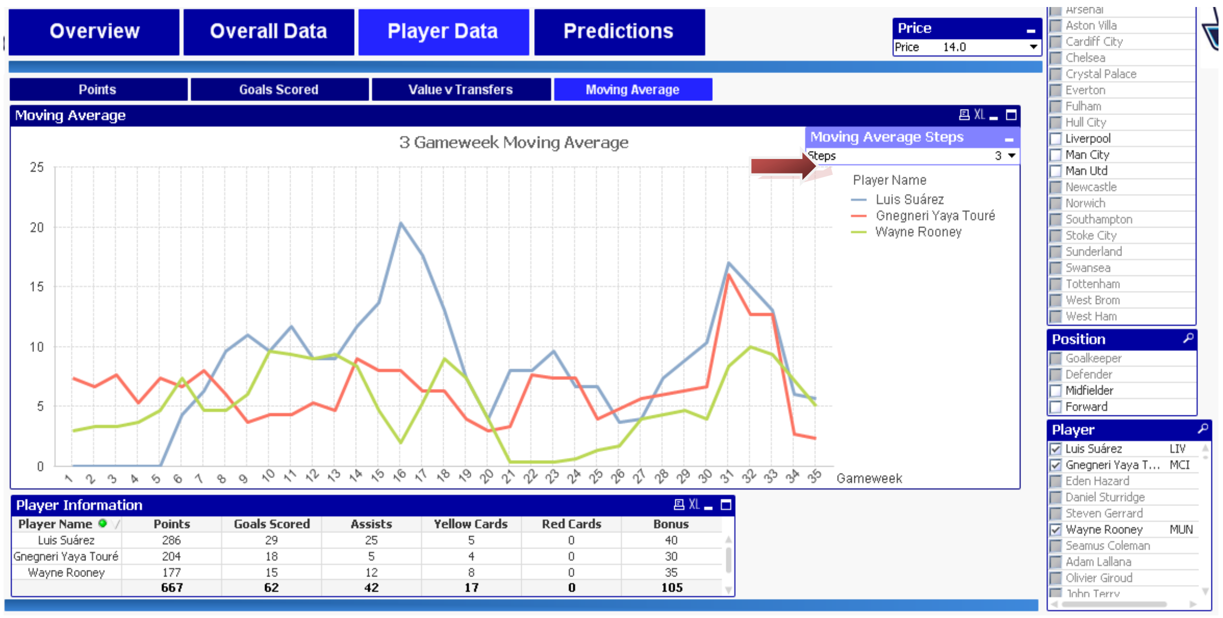


Player Information

Player Name	Points	Goals Scored	Assists	Yellow Cards	Red Cards	Bonus
Luis Suárez	286	29	25	5	0	40
	286	29	25	5	0	40

Fig 5.26: Player data – Price Change per game week vs. Transfers

The moving average tab shows the n-day moving average for each player selected. The players can be compared against each other to provide useful information on the form of certain players. The number of days used to calculate the moving average can be chosen by the user using a drop down box in the graph and a variable.



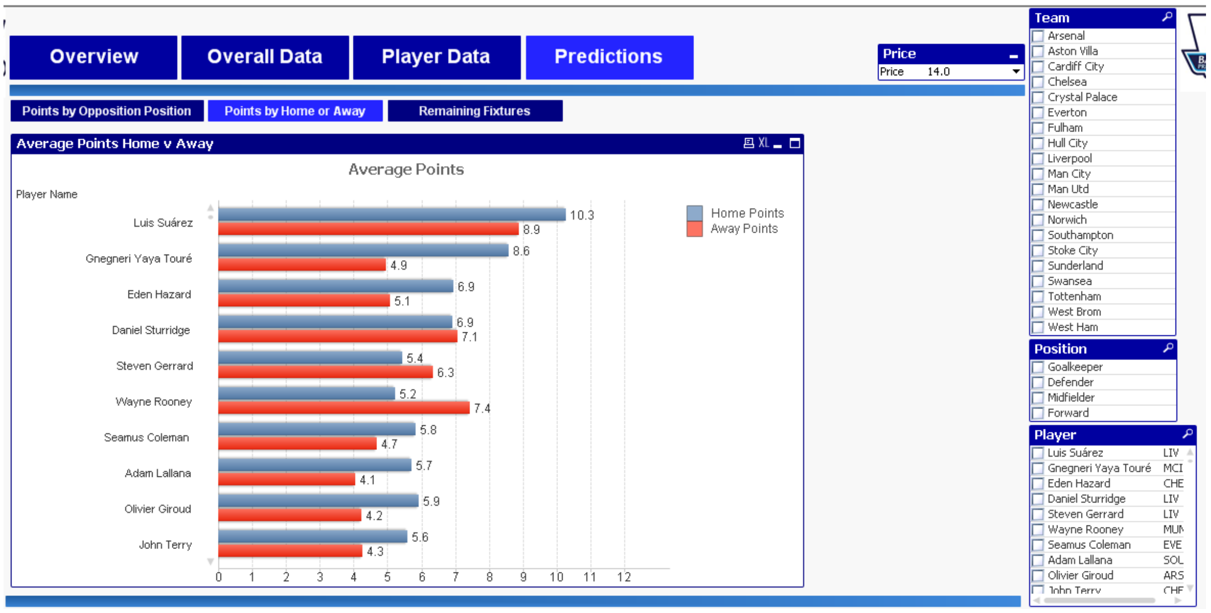
Player Information

Player Name	Points	Goals Scored	Assists	Yellow Cards	Red Cards	Bonus
Luis Suárez	286	29	25	5	0	40
Gnegneri Yaya Touré	204	18	5	4	0	30
Wayne Rooney	177	15	12	8	0	35
	667	**62**	**42**	**17**	**0**	**105**

Fig 5.27: Moving Average Points per Game week

5.9.1.4 Predictions

The predictions tab contains data on the breakout of the player scoring by home vs. Away away matches, top teams vs bottom teams and points scored against teams remaining. The tab home vs. Away gives the user an insight into the average scoring of the player for games at home compared to games played away from home.



Player Name	Home Points	Away Points
Luis Suárez	10.3	8.9
Gnegneri Yaya Touré	8.6	4.9
Eden Hazard	6.9	5.1
Daniel Sturridge	6.9	7.1
Steven Gerrard	5.4	6.3
Wayne Rooney	5.2	7.4
Seamus Coleman	5.8	4.7
Adam Lallana	5.7	4.1
Olivier Giroud	5.9	4.2
John Terry	5.6	4.3

Fig 5.28: Average Points Home vs. Away

As expected, the majority of players have a higher scoring average when playing at home compared to away from home. The sub tab Points by Opposition Position shows the players average scoring against the top n teams compared to the bottom 20 – n teams. The value of n is selected from a drop down box and is stored as a variable. This allows the user to compare average scoring against top teams vs. Bottom teams.

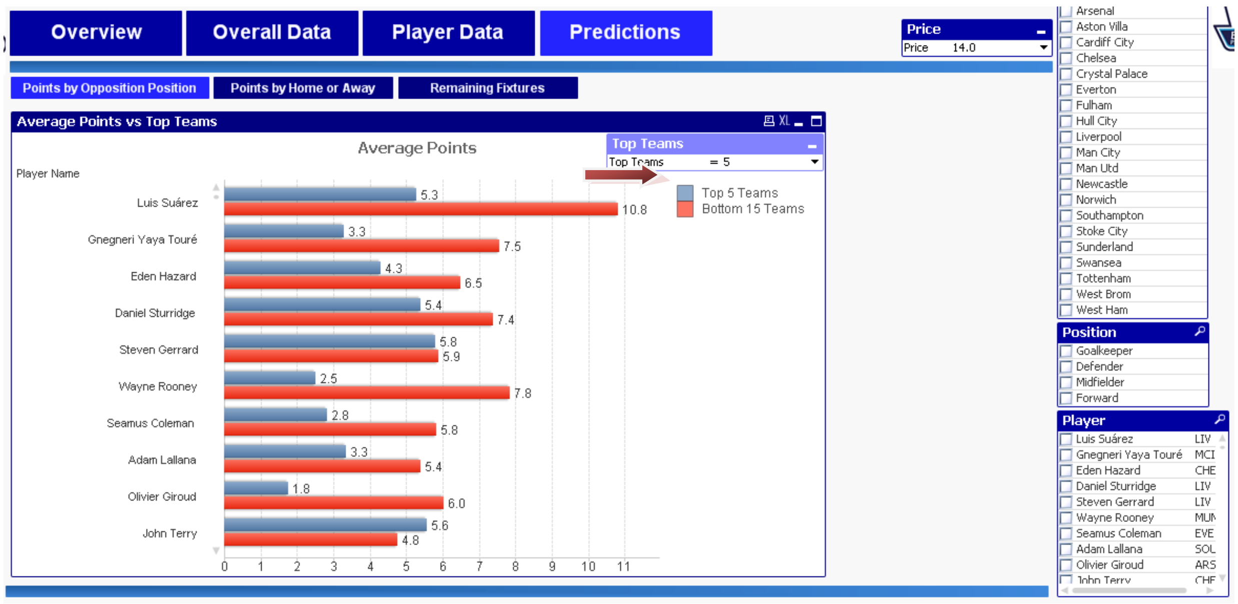


Fig 5.29: Average Points Top vs. Bottom teams

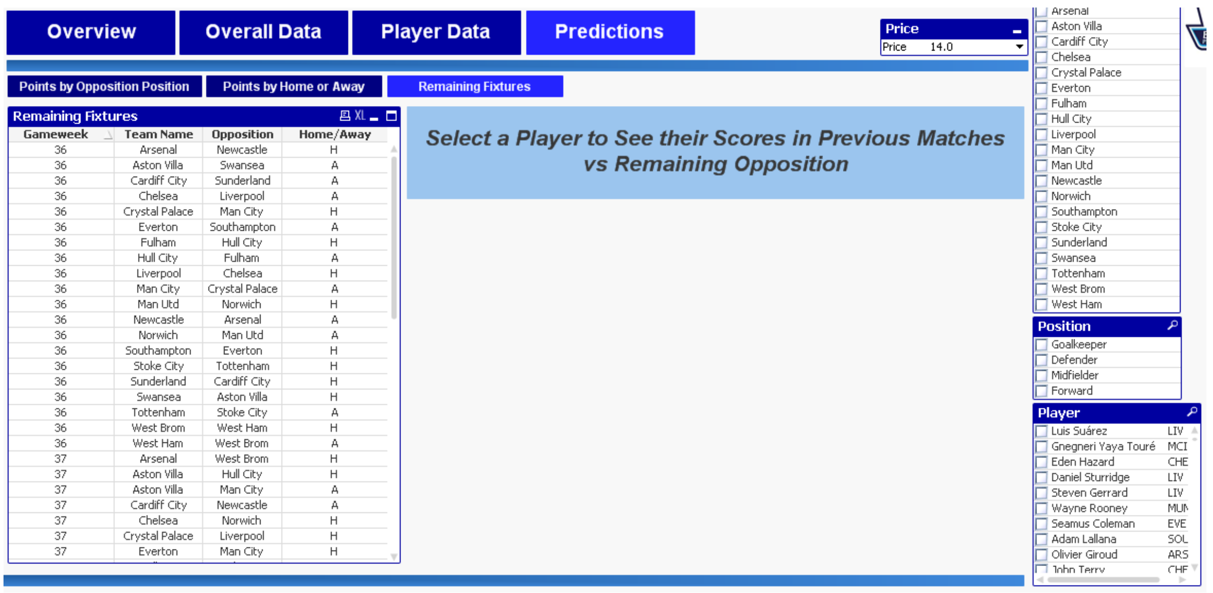
The code for the expression here is given as:

```
=avg(
{<
MINS_PLYD -= {0},
OPPOSITION_POSITION = {"<=$(=vTopTeams)"}
>}
POINTS
)
```

And the code for the dimensions is:

```
=if(PRICE <= vPriceFilter, PLAYER_NAME)
```

The final sub tab in the dashboard is called Remaining Fixtures. Here the user has the ability to see what fixtures a team/player have remaining and the previous scores that player achieved against those teams in the past. This, coupled with the previous two sub-tabs in this sheet, gives the user some valuable insight into the scoring potential of the player for their remaining games.

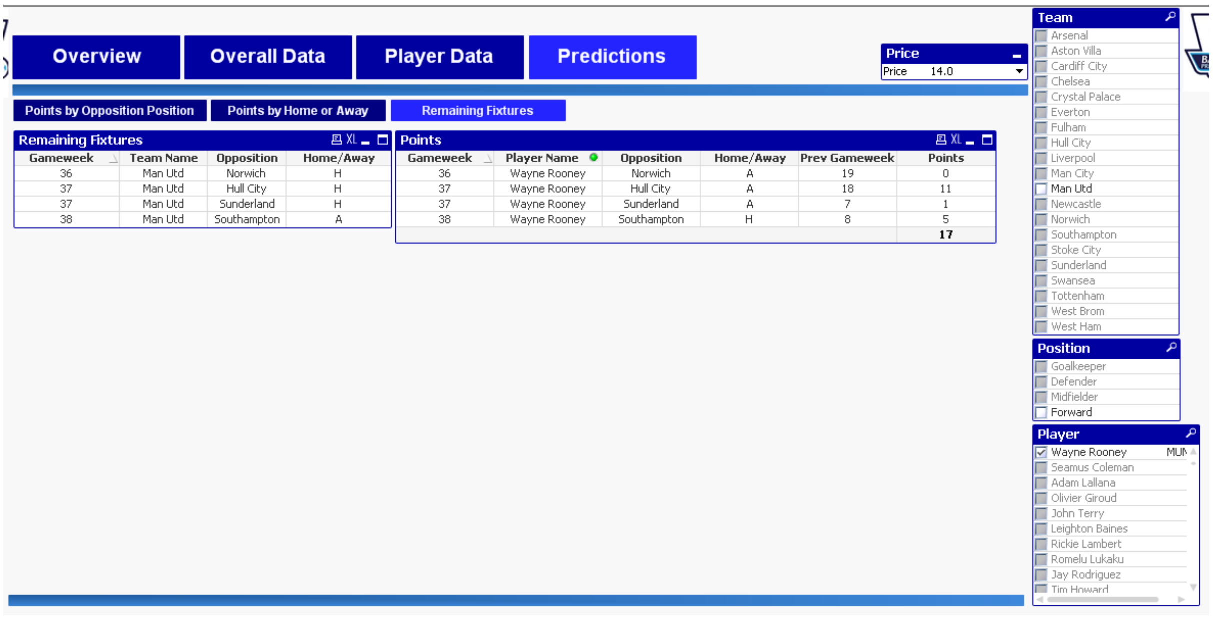


Select a Player to See their Scores in Previous Matches vs Remaining Opposition

Remaining Fixtures

Gameweek	Team Name	Opposition	Home/Away
36	Arsenal	Newcastle	H
36	Aston Villa	Swansea	A
36	Cardiff City	Sunderland	A
36	Chelsea	Liverpool	A
36	Crystal Palace	Man City	H
36	Everton	Southampton	A
36	Fulham	Hull City	H
36	Hull City	Fulham	A
36	Liverpool	Chelsea	H
36	Man City	Crystal Palace	A
36	Man Utd	Norwich	H
36	Newcastle	Arsenal	A
36	Norwich	Man Utd	A
36	Southampton	Everton	H
36	Stoke City	Tottenham	H
36	Sunderland	Cardiff City	H
36	Swansea	Aston Villa	H
36	Tottenham	Stoke City	A
36	West Brom	West Ham	H
36	West Ham	West Brom	A
37	Arsenal	West Brom	H
37	Aston Villa	Hull City	H
37	Aston Villa	Man City	A
37	Cardiff City	Newcastle	A
37	Chelsea	Norwich	H
37	Crystal Palace	Liverpool	H
37	Everton	Man City	H

Fig 5.30: Remaining Fixtures per team



Remaining Fixtures

Gameweek	Team Name	Opposition	Home/Away
36	Man Utd	Norwich	H
37	Man Utd	Hull City	H
37	Man Utd	Sunderland	H
38	Man Utd	Southampton	A

Points

Gameweek	Player Name	Opposition	Home/Away	Prev Gameweek	Points
36	Wayne Rooney	Norwich	A	19	0
37	Wayne Rooney	Hull City	A	18	11
37	Wayne Rooney	Sunderland	A	7	1
38	Wayne Rooney	Southampton	H	8	5
					17

Fig 5.31: Points vs. Remaining teams per Player

5.10 Testing

The testing of the dashboard data was done in phases after each additional piece of functionality was introduced. Once a new version of the dashboard was created regression testing was carried out on the system to ensure that the new features had not broken any of the existing logic. One of the key issues with testing the data in qlikview is ensuring the associations work as required. This involves ensuring that all key fields in the tables are named the same and any fields which are not related are given different field names.

The regression test plan is shown in Appendix H.

6. Further Developments

Given the tight time frame for the project completion, a limited amount of functionality was inserted into the dashboard to ensure that the deadlines in the Proposal Reports were met (Gibbons, 2014). There is a lot of scope remaining in this area for further development both in terms of functionality of the dashboard and in terms of deployment across multiple channels (mobile, internet platforms)

One aspect of the original project requirements specification which was not implemented was the introduction of a data mining program which would take the player scoring data and a number of other independent variables as input and create a model which would predict future scoring based on these variables (Gibbons, 2014). There were a number of reasons why this was not added to this release of the system.

1. The data mining class was being run in parallel with the project and as such not all necessary skills had been covered.
2. The tight time frame of the project meant that other requirements had to be prioritised and this was rated as a "Would" on the MoScoW analysis scale of importance
3. The amount of data available at the time was limited. The data scraping activities began with around 20 game weeks of the season completed which meant there were only 20 rows per player available. This was deemed to not be enough data to complete a worthwhile data mining exercise. However, for the release of the system for next season all of the data from this season will be available so this functionality can be implemented with the R programming language the most likely technology to be used.

As there are few other systems online which carry out the same type of analysis covered here, the next step of development is to host the dashboard online. There are a couple of options available to implement this. The dashboard could be recreated in the tableau software as all documents created in tableau are available online to members of the public (Tableau, 2014). The other option is to purchase server space and deploy the qlikview dashboard to the server. Given the limited budget available to the project the most likely source of action will be to re-create the dashboard in Tableau. This could also prove beneficial or the data mining functionality as Tableau has a built in interface with R.

6.1 Contingencies

The current system is tested and passed for the FPL website as it currently stands. However, the layout of the website is changing from year to year. This means that the scraping programs written for the current website may not be applicable to the website after changes have been made. A full system regression test will be required at the start of next season to identify any changes that need to be made to the program.

In the case that the website changes during the course of this season, a copy is kept of all scripts, text files, databases and dashboards both on the local machine and in the cloud to Google Drive (Google, 2014). This ensures that if the layout of the dashboard does change, the majority of the data from the season will still be available and the project can still complete on time with the caveat that a further development cycle is required for the website layout.

7. Conclusions

The end goal of this project was to create a fully functional, interactive dashboard which managers in the FPL game could use intuitively to help them with decisions about their teams. This goal has been achieved and the dashboard created is a robust tool for any managers in the game. The overall system uses a number of tools which are integrated seamlessly and interact with each other at different steps of the process.

The system is a unique source for all information regarding the FPL game and provides a valuable resource to all users. There is a wide scope for further development of the system and new functionality will be added in the coming months. The summer period, during which there are no premier league games, will provide the time to introduce and test new functionality into the dashboard. The final end goal of having a prediction model in place for next season will add a unique layer of insight which has never before been achieved.

Once all of these features have been implemented and tested, the dashboard will be hosted online and advertised originally through social media. The number of hits the website gets will be tracked and as the word spreads about the tool we hope to see a large increase in traffic to the site. This would hopefully end up with monetisation of the site in terms of advertising revenue.

Bibliography

Crummy, 2013. *Crummy BeautifulSoup.* [Online]
Available at: http://www.crummy.com/software/BeautifulSoup/bs4/doc/
[Accessed 1 March 2014].

DB Engines, 2013. *Database Systems Rankings.* [Online]
Available at: http://db-engines.com/en/ranking
[Accessed 1 May 2014].

Gibbons, B., 2014. *Project Proposal - Extract and Analyse Player Performance Data from the Fantasy Premierleague Website,* Dublin: s.n.

Gibbons, B., 2014. *Requirements Specification - Extract and Analyse Player Performance Data from the Fantasy Premierleague Website,* Dublin: s.n.

Google, 2014. *Google Drive.* [Online]
Available at: https://drive.google.com/
[Accessed 5 Jan 2014].

J6 Design, 2012. *j6 Design.* [Online]
Available at: http://www.j6design.com.au/ClientArea/Whyuseimages
[Accessed 20 April 2014].

MySQL, 2014. *Mysql.* [Online]
Available at: http://dev.mysql.com/downloads/mysql/
[Accessed 30 September 2013].

MySQL, 2014. *Mysql Dev.* [Online]
Available at: http://dev.mysql.com/downloads/connector/odbc/
[Accessed 5 February 2014].

Notepad ++, 2014. *Notepad ++.* [Online]
Available at: http://notepad-plus-plus.org/
[Accessed 21 April 2014].

PLFantasy, 2014. *PLFantasy.* [Online]
Available at: http://premierleaguefantasy.blogspot.ie/p/player-dashboard.html
[Accessed 20 April 2014].

Premier League, 2014. *Fantasy Premier League.* [Online]
Available at: http://fantasy.premierleague.com/
[Accessed 20 April 2014].

Premier League, 2014. *Fantasy Premier League.* [Online]
Available at: http://fantasy.premierleague.com/
[Accessed 23 February 2014].

Premier League, 2014. *Fantasy Premier League API.* [Online]
Available at: http://fantasy.premierleague.com/web/api/elements/180
[Accessed 25 February 2014].

Premier League, 2014. *Fantasy Premier League Rules.* [Online]
Available at: http://fantasy.premierleague.com/rules/
[Accessed 20 April 2014].

Python for Beginners, 2014. *Python API and JSON.* [Online]
Available at: http://www.pythonforbeginners.com/api/python-api-and-json
[Accessed 1 May 2014].

Python, 2014. *Python unicode.* [Online]
Available at: https://docs.python.org/2/howto/unicode.html
[Accessed 1 February 2014].

Python, 2014. *Urllib.* [Online]
Available at: http://docs.python.org/2/library/urllib.html
[Accessed 25 March 2014].

Python, 2014. *urllib python.* [Online]
Available at: https://docs.python.org/2/library/urllib.html
[Accessed 1 May 2014].

Python, 2014. *What is Python.* [Online]
Available at: https://docs.python.org/2/faq/general.html#what-is-python
[Accessed 21 April 2014].

Qlik, 2012. *Qlikview.* [Online]
Available at: qlikview.com
[Accessed 15 March 2014].

Qlik, 2014. *Qlik.* [Online]
Available at: http://www.qlik.com/us/explore/products/overview
[Accessed 1 April 2014].

Qlikview, 2014. *Qlikview Community.* [Online]
Available at: http://community.qlik.com/content
[Accessed 10 April 2014].

Six Feet Up, 2013. *Six Feet Up.* [Online]
Available at: http://www.sixfeetup.com/blog/why-we-choose-python
[Accessed 21 April 2014].

Tableau, 2014. *Tableau.* [Online]
Available at: http://public.tableausoftware.com/views/PlayerPage/Dashboard1?:showVizHome=no
[Accessed 20 April 2014].

Tableau, 2014. *Tableau.* [Online]
Available at: http://www.tableausoftware.com/products/online
[Accessed 3 May 2014].

Visual Intellegence, 2014. *Why Choose Qlikview.* [Online]
Available at: http://www.visualintelligence.co.nz/QlikView.php
[Accessed 1 May 2014].

Wiki Python, 2014. *Python 2 or Python 3.* [Online]
Available at: https://wiki.python.org/moin/Python2orPython3
[Accessed 21 April 2014].

Appendices

Appendix A – Project Proposal

Objectives and Contribution to the Knowledge

The Premier League Fantasy Football game is played by over 3 million people worldwide. The unique User Interface allows players to easily access their team scores, make transfers, check their mini-leagues and make substitutions.

The project idea is to create a user friendly dashboard which will give detailed stats on each players scoring throughout the season allowing them to make informed decisions on transfers, substitutions etc.

Background

All of the data is provided on the website as text files but no graphical representation of this data can be generated. For a player to check certain stats across a team/player/position they would have to copy and paste the data they want to an excel file and generate whatever results they want themselves. This would be extremely time-consuming and would have to be repeated after every gameweek has completed.

With the final dashboard, the idea is that all of this information will be consolidated into one master view and the user can slice and dice the data whatever way they want with no need for manual calculation.

The concept of a dashboard for fantasy football stats does exist but in most cases membership to a website and, in a lot of cases, a fee is required. One website that creates a free dashboard is called

http://premierleaguefantasy.blogspot.ie/p/player-dashboard.html

This is created using Tableau but it is quite cumbersome and not very user friendly.

Technical Approach

There are a number of steps involved in the implementation of the system.

1. Create a script(s) to scrape the website for all of the historical scoring data for each player using python
2. Cleanse the data using python/excel
3. Create a schema for mysql database and insert structured data into tables
4. Use Qlikview to create a dashboard of all stats based on the tables in this database
5. (Possibly) mine the available data to produce suggestions on what players to buy/sell based on historical matches.

Special resources required

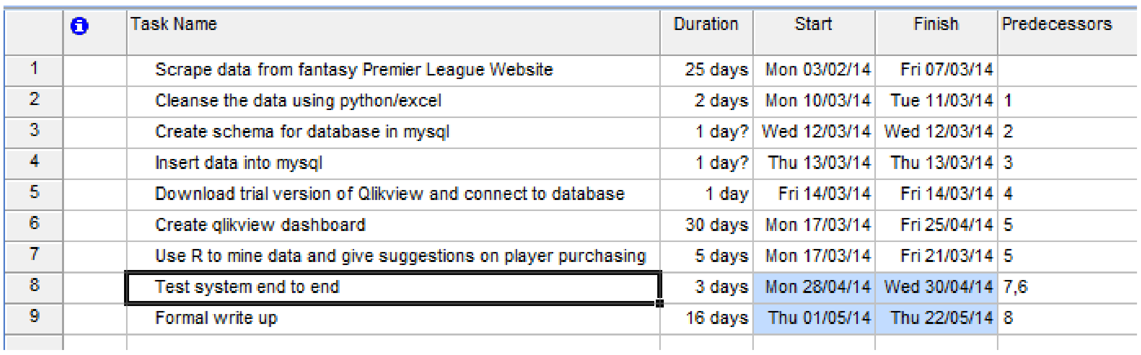
Python (free)

Mysql(free)

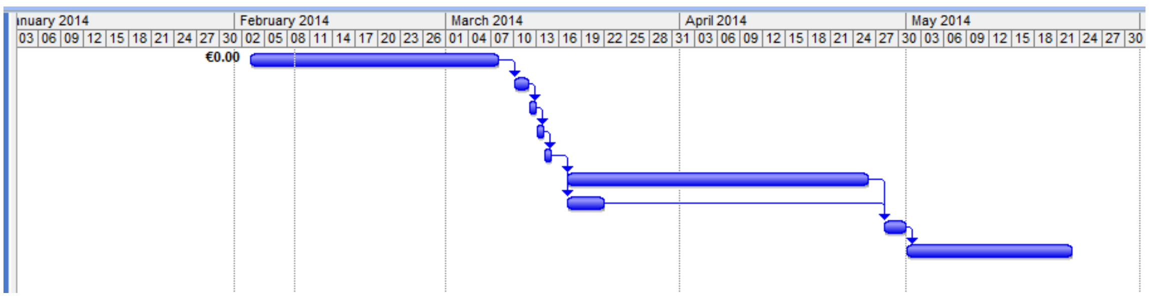
Excel(download limited free version)

Qlikview(download limited free personal edition)

Project Plan



		Task Name	Duration	Start	Finish	Predecessors
1		Scrape data from fantasy Premier League Website	25 days	Mon 03/02/14	Fri 07/03/14	
2		Cleanse the data using python/excel	2 days	Mon 10/03/14	Tue 11/03/14	1
3		Create schema for database in mysql	1 day?	Wed 12/03/14	Wed 12/03/14	2
4		Insert data into mysql	1 day?	Thu 13/03/14	Thu 13/03/14	3
5		Download trial version of Qlikview and connect to database	1 day	Fri 14/03/14	Fri 14/03/14	4
6		Create qlikview dashboard	30 days	Mon 17/03/14	Fri 25/04/14	5
7		Use R to mine data and give suggestions on player purchasing	5 days	Mon 17/03/14	Fri 21/03/14	5
8		Test system end to end	3 days	Mon 28/04/14	Wed 30/04/14	7,6
9		Formal write up	16 days	Thu 01/05/14	Thu 22/05/14	8



Technical Details

The principle language used will be python and the urllib, regex and json libraries will be used to scrape the data from the website. SQL will be used to manipulate the data also in mysql with the final dashboard created using qlikview after making a connection with the sql database.

Analysis may be done with R further down the line in the project to mine the available data whereby the data will be mined to give suggestions on which players to buy/sell based on their historical performances against teams in the division.

Systems/Datasets

The system description will be as per the technical approach. The source data is to be pulled from the fantasy football website itself using mainly the json files which are sent to the server. As the data will be unstructured, a process will be put in place to cleanse all of the data before inputting into a database.There will be a couple of iterations to the extraction of data. At first I will use the regex library to pull the source data from the XML code behind the website. If this is taking too long, which is possible as the amount of code to be scraped in this case will be huge given the number of connections to be made to the website, then I will use the json library to pull all the json files back if they are available.

Evaluation, Tests and Analysis

The system will be tested at every stage and results will be compared back to the original website for comparison purposes. Eg. A check will be done to ensure a players points on a certain game week match the value given on the website. Tests will also be carried out that the data has been cleansed properly and any bugs in the cleansing programme will be rectified as far as possible. Notes will be made of any issues encountered for documentation purposes.

The performance of the system will be measured by having players of the fantasy game use the dashboard and give feedback on the output. Any feedback will be incorporated as far as possible and is feasible into the next iteration of the system.

The main performance metrics which can be measured empirically are the speed of the scraping algorithm and the speed of the queries to the database.

Consultation with Specialisation Person(s)

Oisin Creaner: Voiced reservations about the size of the dataset which will be analysed.

Stephen Mullins: User of the fantasy football website. Thought the idea was a very good one as there is nothing similar readily available to fantasy football players.

Appendix B – Requirements Specification

Introduction

Purpose

The purpose of this document is to set out the requirements for the development of a Qlikview dashboard for the analysis of Fantasy Premier League Data. The document will describe the functional requirements for the user to interact with the dashboard and also how the systems (website -> mysql -> Qlikview) interact with one another. The non-functional requirements such as connectivity, performance and extendibility will also be discussed

The intended customers are the end-users of the dashboard, any individuals assigned to test the system end to end and the project sponsor.

Project Scope

The scope of the project is to develop a dashboard in Qlikview which the end user can interact with in order to gain insight and alternative views into player scoring data. The data used for the project is scraped from the Fantasy Premier League website (Premier League, 2014) using the python scripting language. The python programme will make multiple requests to the website in order to extract all data for every player in the game.

The data will be saved down to a CSV file and loaded into a mysql database environment.

A connection will then be made between Qlikview and the database via ODBC. The data will then be available for analysis in Qlikview. All the graphs and tables will be developed in the Qlikview environment

An informal interview was carried out with a number of Fantasy Premier League users to elicit any functional requirements which they would find useful in helping them maximise their point scoring potential.

The number of users interviewed was limited to those available at the time of elicitation. Many individuals available for interviewing did not play the game and so were not used for the process.

Definitions, Acronyms, and Abbreviations

SQL	Structured Query Language
FPL	Fantasy Premier League
ODBC	Open Database Connectivity
Qlikview	Data Visualisation software for creation of user defined dashboards

User Requirements Definition

The end users interviewed gave very good feedback as to what functionality they would like to see in the system. The main functionality suggested was:

- The user can select a certain player and see multiple views of the players scoring statistics
- The user can see trendlines for player scoring statistics
- The user can interact with the dashboard intuitively
- The information in the dashboard can be exported to excel
- The dashboard can suggest which players to purchase based on a range of input variables (price etc.)

Requirements Specification

Functional requirements

Requirement Number	Requirement	MoSCoW Priority
1	**Web Scraping Programme**	
1.1	Python programme must scrape data from Fantasy Premier League website for all players in the game	M
1.2	Programme must be able to interpret .json file types	M
1.3	Scraping programme must be run after every gameweek has completed and all bonus points updated	M
1.4	Ouput file from scraping programme is to be a single csv file with all player info	M
1.5	Programme must be ran from the command line prompt	M
2	**Mysql Database**	
2.1	The csv file must be loaded into a mysql database	M
2.2	The mysql tablesand data load should be ran together in one script.	S
3	**Qlikview Dashboard**	
3.1	The Qlikview application should be downloaded and connected to the mysql database	M
3.2	The data from the mysql database must be loaded into the qlikview environment using relevant code	M
3.3	Graphs and tables of player statistics and trendlines must be created within the qlikview application. The code for each of these graphs/tables to be generated independently	M
3.4	All of the charts graphs must be exportable to excel	M
3.5	User cannot make amendments to dashboard. Can only view and select the data	M
4.1	A model should be created in R which will interpret input variables for each player and output suggestions based on which players are good/bad picks at that time based on the underlying variables	S

Requirement 1: Create Web Scraping Programme

Description & Priority

This requirement describes how the system scrapes the data from the website. This is essential to the overall system as it is where all of the data is extracted. As such it is a Must on the MoSCoW scale.

Use Case

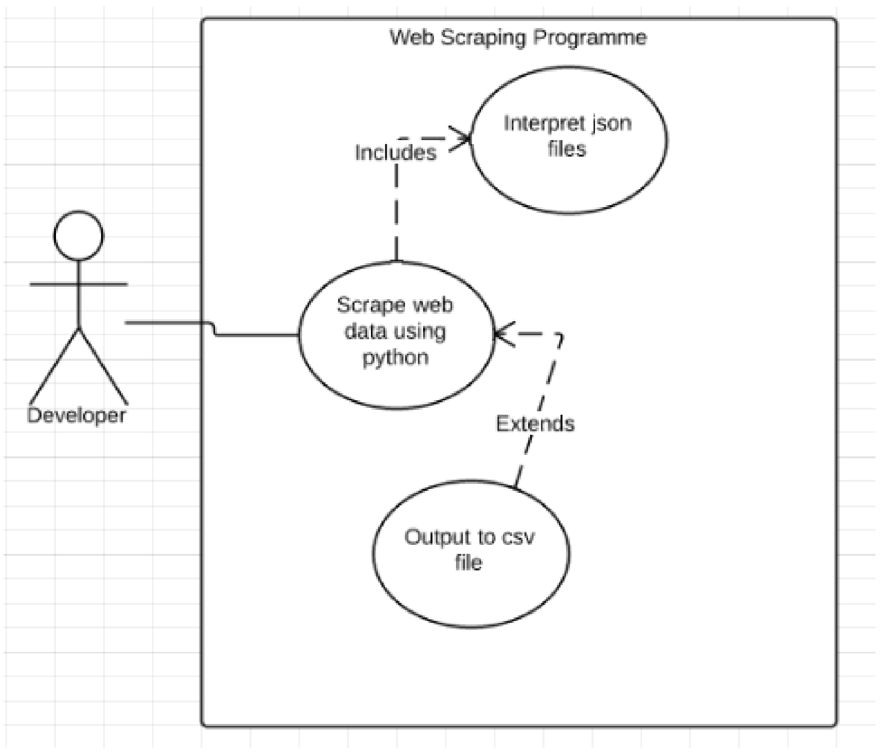
Scope

The scope of this use case is to show the how the system extracts the data from the website

Description

This use case describes the flow of the python system.

Use Case Diagram



Flow Description

Precondition

The correct URL is given for the web API

Activation

This use case starts when the developer runs the code from the command line

Termination

The system terminates when all of the data has been extracted

Post condition

The data for all players is saved down to a csv file

Requirement 2: Load csv data into Mysql Database

Description & Priority

The data from the csv file is loaded into a mysql database for ease of access. This must be done so we can connect the database to the qlikview application.

Use Case: 2.1-2.2

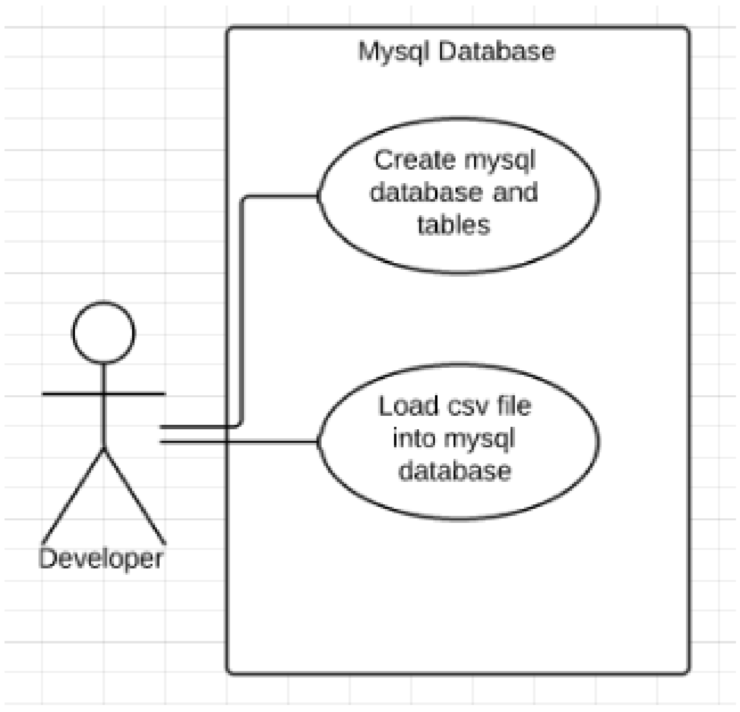
Scope

The scope of this use case is to show how the data is loaded into the mysql database.

Description

This use case describes the mysql database

Use Case Diagram


Mysql Database
Create mysql database and tables
Load csv file into mysql database
Developer

Flow Description

Precondition

The data has been extracted and saved to a csv file

Activation

The mysql script is run from the mysql prompt

Termination

The system is terminated when the script has successfully run

Post condition

The data will be available in the mysql database

Requirement 3: Create Qlikview Dashboard

Description & Priority

The Qlikview dashboard will act as the GUI for the system and enable the user to make selections and view the data

Use Case: 2.1-2.2

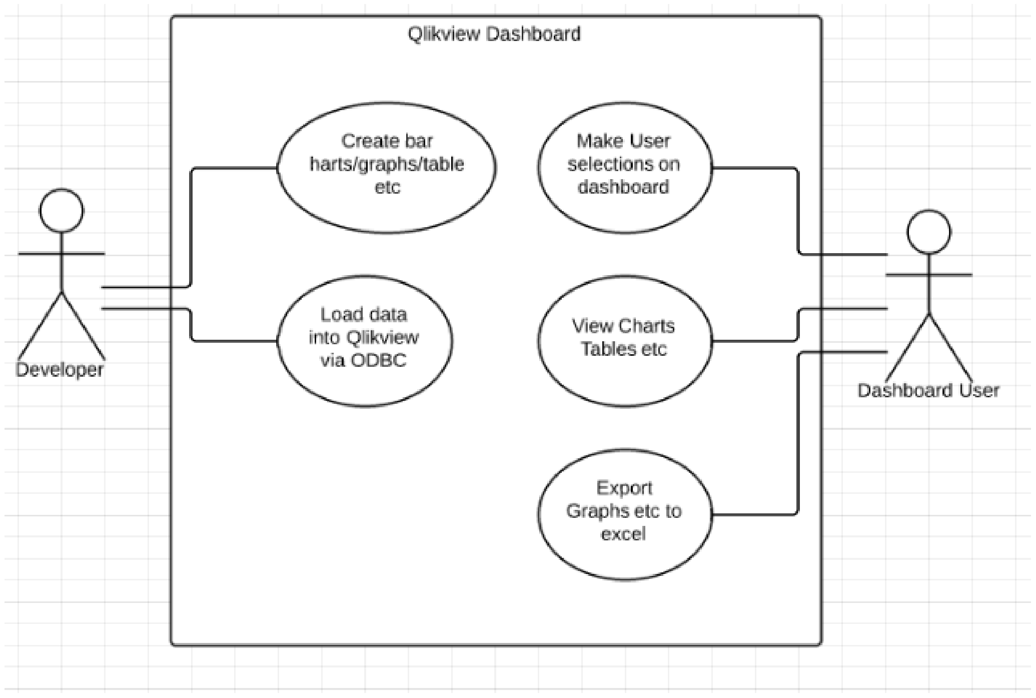
Scope

The scope of this use case is to show how the dashboard is created and how the user interacts with it.

Description

This use case describes the qlikview dashboard

Use Case Diagram


Qlikview Dashboard
Create bar harts/graphs/table etc
Load data into Qlikview via ODBC
Make User selections on dashboard
View Charts Tables etc
Export Graphs etc to excel
Developer
Dashboard User

Flow Description

Precondition

The ODBC connection between the mysql database and qlikview has been established

Activation

The qlikview load script has been run

Termination

When the load script is finished

Post condition

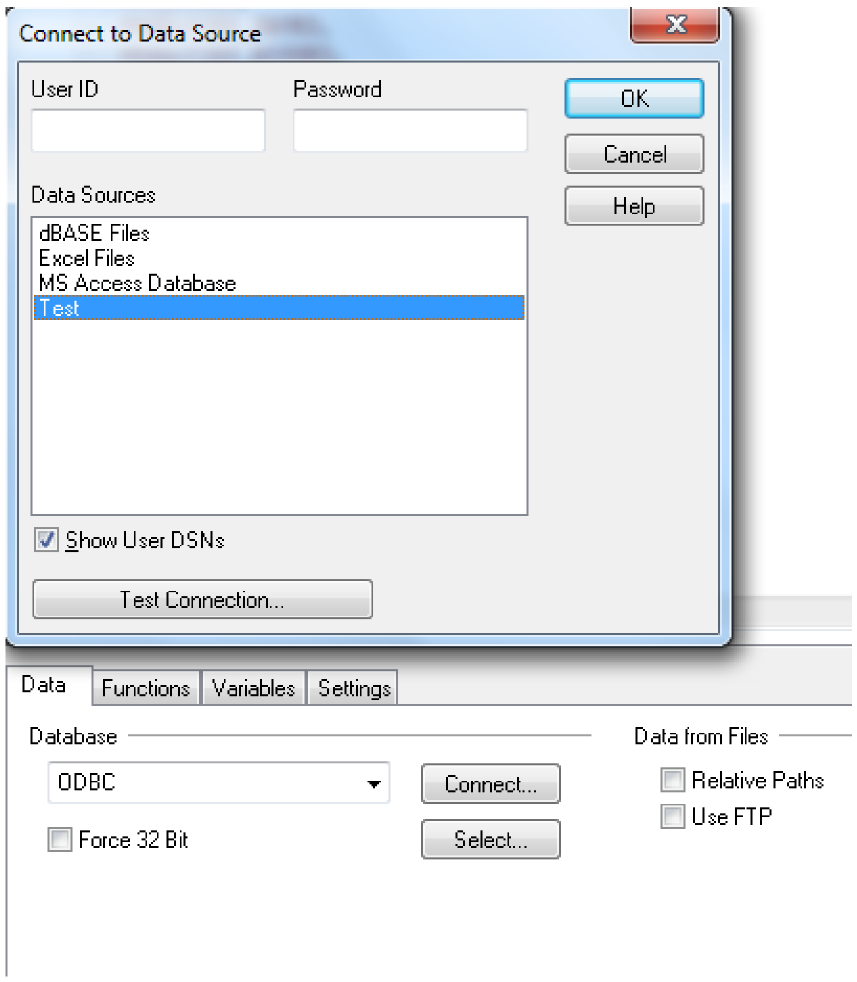
The data will be visible in the dashboard.

Non-Functional Requirements

Connectivity requirement

The python programme must be able to establish a connection with the host website of the fantasy premier league. In order to do this, the package urllib (Python, 2014) will be imported into the programme. This allows the programme to establish a connection with the website and for scraping to commence

A connection must also be made between the qlikview application and the mysql database. In order to do this, an ODBC driver will be downloaded and a connection set up to the mysql database in question.



In this case the database is called Test and the connection is made via ODBC. The DSN was set up in the configuration settings for the ODBC driver.

Recovery requirement

If for some reason the website is down and access is not possible, or the running of the system fails for some reason, a backup copy of all tables in the database will be kept to ensure that data is not lost and tables can be repopulated.

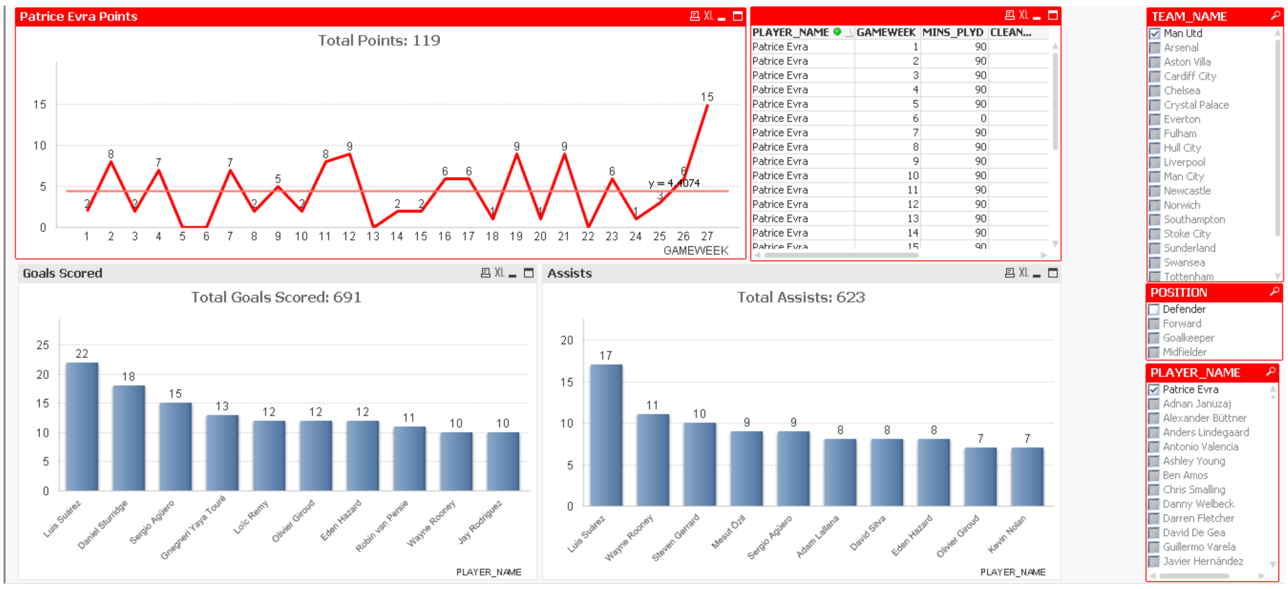
Extendibility requirement

Because the layout and format of a website can change, the programme may not be usable for next season's data. A copy of all of the current season data will be kept so that the analysis can continue into next season when adjustments to the programme may have to be made.

Interface requirements

GUI

The GUI for the system will be a completed dashboard which will contain data for all players on the game. The user will be able to make selections on the data based on a number of different attributes e.g. Goals/assists/points etc. and this can be drilled down to team/player level. The diagram below shows an initial mockup of the dashboard and how the user can make selections:



PLAYER_NAME	GAMEWEEK	MINS_PLYD	CLEAN...
Patrice Evra	1	90	
Patrice Evra	2	90	
Patrice Evra	3	90	
Patrice Evra	4	90	
Patrice Evra	5	90	
Patrice Evra	6	0	
Patrice Evra	7	90	
Patrice Evra	8	90	
Patrice Evra	9	90	
Patrice Evra	10	90	
Patrice Evra	11	90	
Patrice Evra	12	90	
Patrice Evra	13	90	
Patrice Evra	14	90	
Patrice Evra	15	90	

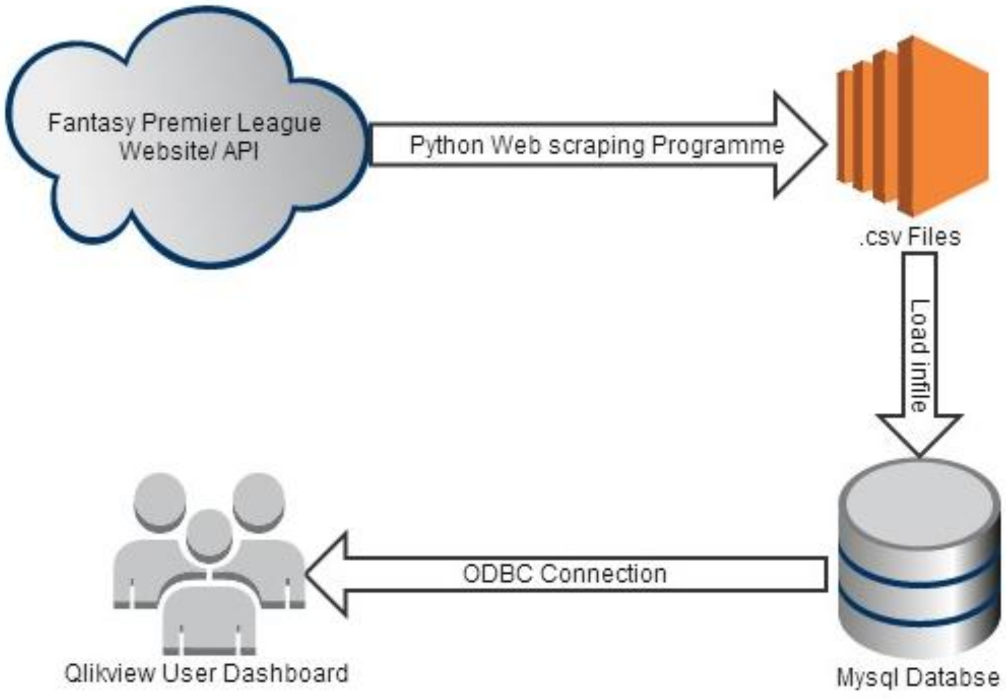
Application Programming Interfaces (API)

The system will use the fantasy premier league's API. An example is shown in the link below:

http://fantasy.premierleague.com/web/api/elements/180/ (Premier League, 2014)

This produces a JSON file which is interperated by the JSON package in python and the relevant data is extracted and saved down to a csv file.

System Architecture



System Evolution

There are a number of add-ons that could be made to the system after the project has completed. There is scope to add additional features such as a data mining algorithm in R which could be run against the data and based on a number of variables the optimum transfer could be generated.

If the system is functioning correctly, there are no major bugs and a beta test by a number of game players was carried out with positive feedback, then it would be possible to host the dashboard online allowing users from all over the world to access the functionality. Considerations such as server cost etc would need to be taken into account for this and it would only be feasible if the system was running efficiently and with no errors.

Appendix D – Progress Management Reports

Progress Management Report 1

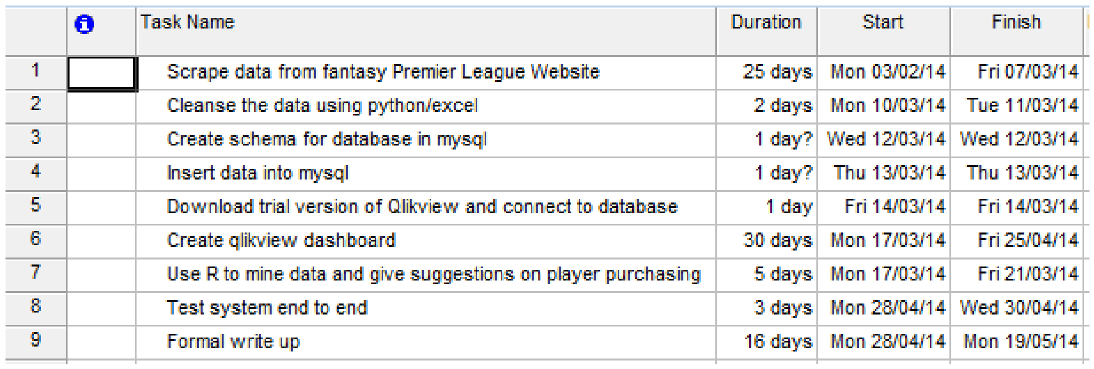
Purpose of Document

The purpose of this document is to give an update to all relevant stakeholders on the project status. This will give the stakeholders a chance to voice their concerns/issues with the progress of the project or any other issues or concerns they may have. The outcome of the report will dictate future activities as, based on discussions, some functionality may be added, removed or adapted.

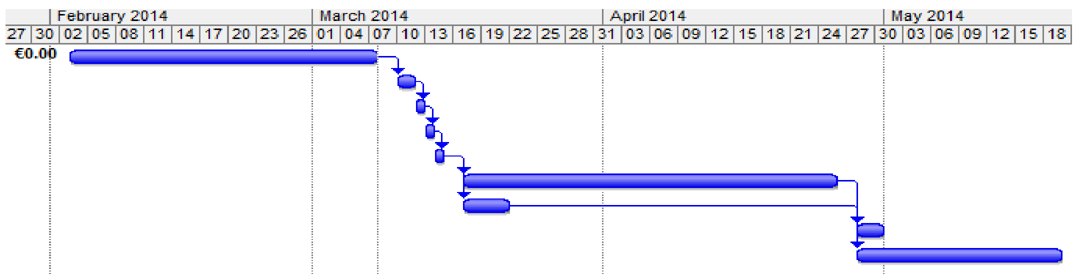
The document should also provide an idea as to the resource requirements for the remainder of the project in order to wait on programme.

Project Timeline

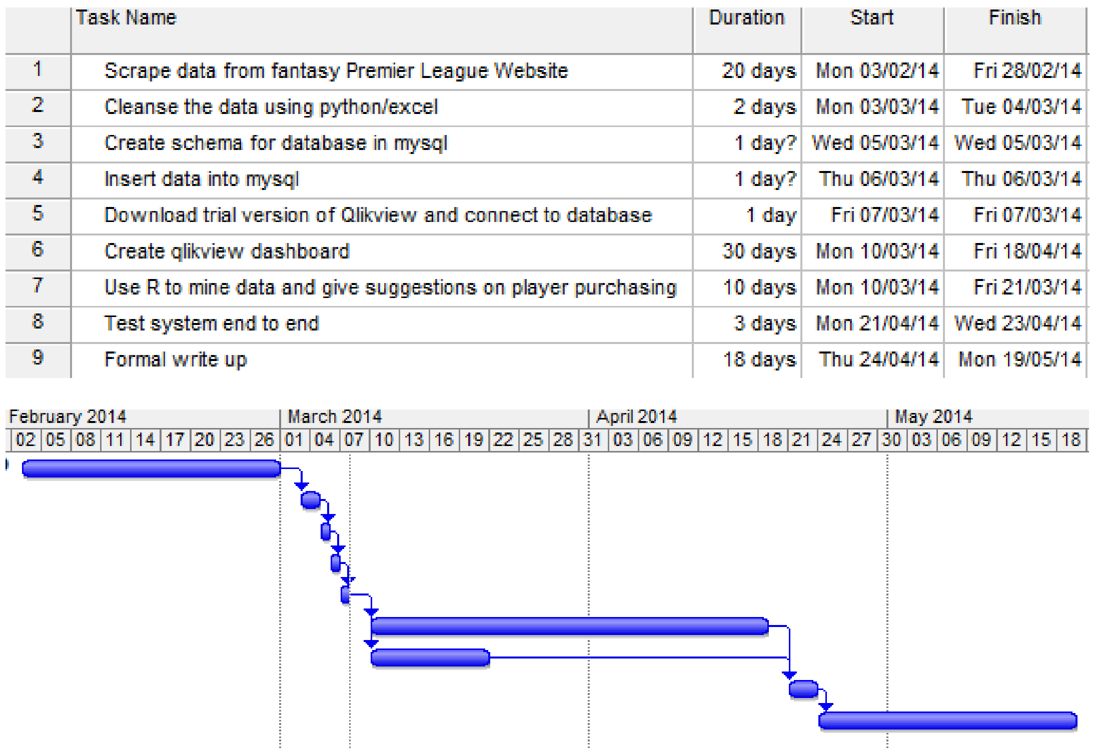
The diagrams below show the original project plan from the Project Proposal document (Gibbons, 2014).



	ℹ	Task Name	Duration	Start	Finish
1		Scrape data from fantasy Premier League Website	25 days	Mon 03/02/14	Fri 07/03/14
2		Cleanse the data using python/excel	2 days	Mon 10/03/14	Tue 11/03/14
3		Create schema for database in mysql	1 day?	Wed 12/03/14	Wed 12/03/14
4		Insert data into mysql	1 day?	Thu 13/03/14	Thu 13/03/14
5		Download trial version of Qlikview and connect to database	1 day	Fri 14/03/14	Fri 14/03/14
6		Create qlikview dashboard	30 days	Mon 17/03/14	Fri 25/04/14
7		Use R to mine data and give suggestions on player purchasing	5 days	Mon 17/03/14	Fri 21/03/14
8		Test system end to end	3 days	Mon 28/04/14	Wed 30/04/14
9		Formal write up	16 days	Mon 28/04/14	Mon 19/05/14



The below diagrams show the updated project plan at this stage in the project:



	Task Name	Duration	Start	Finish
1	Scrape data from fantasy Premier League Website	20 days	Mon 03/02/14	Fri 28/02/14
2	Cleanse the data using python/excel	2 days	Mon 03/03/14	Tue 04/03/14
3	Create schema for database in mysql	1 day?	Wed 05/03/14	Wed 05/03/14
4	Insert data into mysql	1 day?	Thu 06/03/14	Thu 06/03/14
5	Download trial version of Qlikview and connect to database	1 day	Fri 07/03/14	Fri 07/03/14
6	Create qlikview dashboard	30 days	Mon 10/03/14	Fri 18/04/14
7	Use R to mine data and give suggestions on player purchasing	10 days	Mon 10/03/14	Fri 21/03/14
8	Test system end to end	3 days	Mon 21/04/14	Wed 23/04/14
9	Formal write up	18 days	Thu 24/04/14	Mon 19/05/14

As we can see, step 1 in the project has taken 5 less days to complete than originally planned (25 days – 20 days). This 5 day accumulated float has been built into the data mining and report writing activities.

Project Requirements Status

The below chart indicates the status of each of the current requirements. The status will be green if it is on schedule, amber if there is a risk of it going off schedule and red if it is currently off schedule. This information was taken from the Requirements Specification (Gibbons, 2014)

Requirement Number	**Requirement**	**% Complete**	**Status**
1	**Web Scraping Programme**		
1.1	Python programme must scrape data from Fantasy Premier League website for all players in the game	100	🟩
1.2	Programme must be able to interpret .json file types	100	🟩
1.3	Scraping programme must be run after every gameweek has completed and all bonus points updated	100	🟩
1.4	Ouput file from scraping programme is to be a single csv file with all player info	100	🟩
1.5	Programme must be ran from the command line prompt	100	🟩

2	**Mysql Database**		
2.1	The csv file must be loaded into a mysql database	100	🟩
2.2	The mysql tablesand data load should be ran together in one script.	100	🟩
3	**Qlikview Dashboard**		
3.1	The Qlikview application should be downloaded and connected to the mysql database	100	🟩
3.2	The data from the mysql database must be loaded into the qlikview environment using relevant code	100	🟩
3.3	Graphs and tables of player statistics and trendlines must be created within the qlikview application. The code for each of these graphs/tables to be generated independently	10	🟨
3.4	All of the charts graphs must be exportable to excel	5	🟨
3.5	User cannot make amendments to dashboard. Can only view and select the data	0	🟨
4.1	A model should be created in R which will interpret input variables for each player and output suggestions based on which players are good/bad picks at that time based on the underlying variables	0	🟨

Planned Actions

Planned Actions	Importance
Complete the Qlikview dashboard	High
Regression testing of all previous functionality after every run	High
Test system end to end	High
Analyse the incoming data in R and link to the application	Medium
Update progress charts and RAID logs	Low

RAID Log

Risks

Risk Ref	Risk Category	Risk Description	Raised by	Date Identified	Priority	Impact	Prob	Mitigation Category	Mitigation	Owner
R01	technology	No data backup available	BG	01-Mar-14	H	H	L	prevention	Backup all files to Google Drive	BG
R02	Learning	Not enough covered in data mining class to adapt R functionality	BG	01-Mar-14	M	M	H	acceptance	Try to learn enough techniques in own time	BG
R03	Learning	Expiry of free trial software	BG	01-Mar-14	M	M	L	contingency	Work in college instead of home	BG

Assumptions

Ref #	Assumption	Importance	Certainty	Influence	Test	Test Date
A01	Supervisors will provide prompt feedback and guidance	4 - critical	3 - Probable	H	Send request to test level of response	26-Feb-14
A02	No major injuries/illness will be incurred by developer	4 - critical	1 - unknown	H	N/A	-
A03	College facilities will be available throughout the lifecycle of the porject	3 - important	3 - Probable	H	Send an email to facilities to confirm	07-Mar-14

Closed Issues

Issue Ref	Issue Description	Raised by	Date Raised	Impact	Priority	Action Plan	Status	Owner	Target Resolution Date	Actual Resolution Date
101	Error thrown when running the python script	BG	27-Feb-14	H	H	Identify the issue using a try-except block in python script	closed	BG	07-Mar-14	04-Mar-14
102	Not all players were being added to the data file	BG	30/2/14	H	H	Output all of the players who were not being returned to a seperate error file and check for any consistencies	closed	BG	10-Mar-14	07-Mar-14
103	Error thrown when loading txt file into mysql	BG	30/2/14	H	H	Search internet for cause of issue	closed	BG	10-Mar-14	07-Mar-14

Dependencies

Dependency Ref	Project	Dependency Description	Raised by	Date Raised	Impact	Priority	Period Affected	Action Plan	Owner	Target Resolution Date	Actual Resolution Date
D001	Company Facilities	IT facilities available for running model	BG	26-Feb-14	M	H	Feb - May	Confirm availability with IT	BG	May-14	May-14

Conclusions and Suggestions

At this moment in time the project is well on schedule to be fully completed by the set deadline. Any risks or issues outlined in this document should be monitored on an ongoing basis throughout the project to ensure they do not adversely affect the completion date. The RAID log should be kept fully up to date to enable the effective management of risks and issues and these should then be assigned to the appropriate stakeholder and addressed as appropriate.

No additional resources are required for the project at this time.

Progress Management Report 2

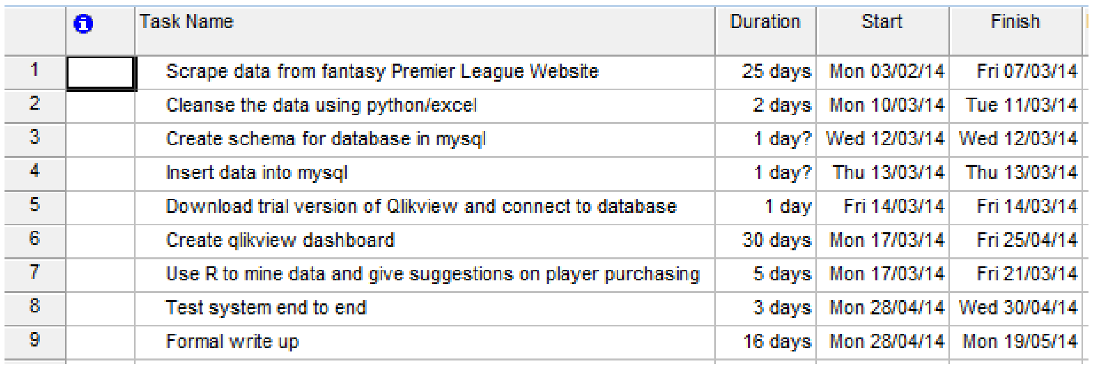
Purpose of Document

The purpose of this document is to give an update to all relevant stakeholders on the project status. This will give the stakeholders a chance to voice their concerns/issues with the progress of the project or any other issues or concerns they may have. The outcome of the report will dictate future activities as, based on discussions, some functionality may be added, removed or adapted.

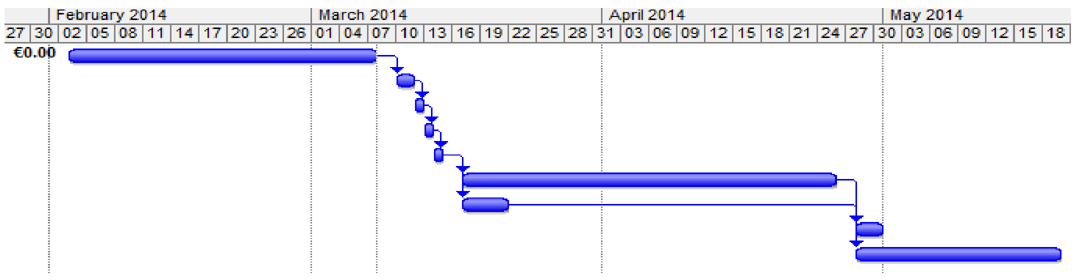
The document should also provide an idea as to the resource requirements for the remainder of the project in order to wait on programme.

Project Timeline

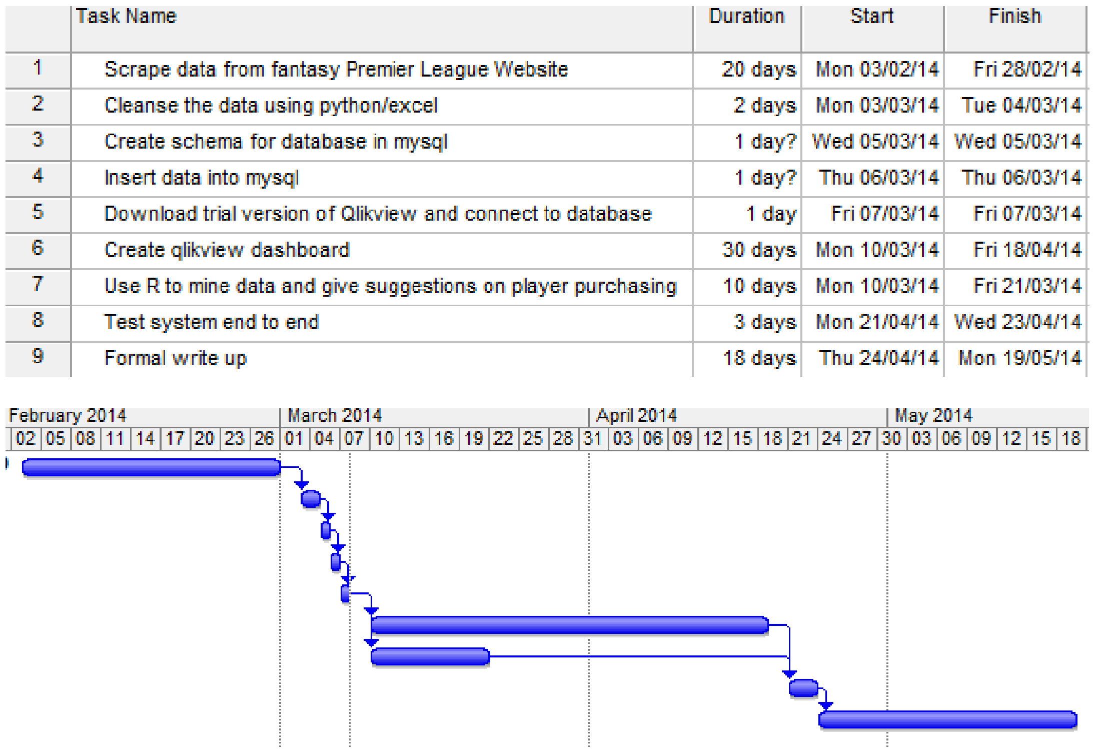
The diagrams below show the original project plan from the Project Proposal document (Gibbons, 2014).



	ℹ	Task Name	Duration	Start	Finish
1		Scrape data from fantasy Premier League Website	25 days	Mon 03/02/14	Fri 07/03/14
2		Cleanse the data using python/excel	2 days	Mon 10/03/14	Tue 11/03/14
3		Create schema for database in mysql	1 day?	Wed 12/03/14	Wed 12/03/14
4		Insert data into mysql	1 day?	Thu 13/03/14	Thu 13/03/14
5		Download trial version of Qlikview and connect to database	1 day	Fri 14/03/14	Fri 14/03/14
6		Create qlikview dashboard	30 days	Mon 17/03/14	Fri 25/04/14
7		Use R to mine data and give suggestions on player purchasing	5 days	Mon 17/03/14	Fri 21/03/14
8		Test system end to end	3 days	Mon 28/04/14	Wed 30/04/14
9		Formal write up	16 days	Mon 28/04/14	Mon 19/05/14



The below diagrams show the updated project plan at this stage in the project:



	Task Name	Duration	Start	Finish
1	Scrape data from fantasy Premier League Website	20 days	Mon 03/02/14	Fri 28/02/14
2	Cleanse the data using python/excel	2 days	Mon 03/03/14	Tue 04/03/14
3	Create schema for database in mysql	1 day?	Wed 05/03/14	Wed 05/03/14
4	Insert data into mysql	1 day?	Thu 06/03/14	Thu 06/03/14
5	Download trial version of Qlikview and connect to database	1 day	Fri 07/03/14	Fri 07/03/14
6	Create qlikview dashboard	30 days	Mon 10/03/14	Fri 18/04/14
7	Use R to mine data and give suggestions on player purchasing	10 days	Mon 10/03/14	Fri 21/03/14
8	Test system end to end	3 days	Mon 21/04/14	Wed 23/04/14
9	Formal write up	18 days	Thu 24/04/14	Mon 19/05/14

Note that the project schedule has not altered since the last report

Project Requirements Status

The below chart indicates the status of each of the current requirements. **The status will be green if it is on schedule, amber if there is a risk of it going off schedule and red if it is currently off schedule.** This information was taken from the Requirements Specification (Gibbons, 2014)

Requirement Number	**Requirement**	**% Complete**	**Status**
1	**Web Scraping Programme**		
1.1	Python programme must scrape data from Fantasy Premier League website for all players in the game	100	🟩
1.2	Programme must be able to interpret .json file types	100	🟩
1.3	Scraping programme must be run after every gameweek has completed and all bonus points updated	100	🟩

1.4	Ouput file from scraping programme is to be a single csv file with all player info	100	🟩
1.5	Programme must be ran from the command line prompt	100	🟩
2	**Mysql Database**		
2.1	The csv file must be loaded into a mysql database	100	🟩
2.2	The mysql tablesand data load should be ran together in one script.	100	🟩
3	**Qlikview Dashboard**		
3.1	The Qlikview application should be downloaded and connected to the mysql database	100	🟩
3.2	The data from the mysql database must be loaded into the qlikview environment using relevant code	100	🟩
3.3	Graphs and tables of player statistics and trendlines must be created within the qlikview application. The code for each of these graphs/tables to be generated independently	50	🟨
3.4	All of the charts graphs must be exportable to excel	50	🟨
3.5	User cannot make amendments to dashboard. Can only view and select the data	0	🟨

4.1	A model should be created in R which will interpret input variables for each player and output suggestions based on which players are good/bad picks at that time based on the underlying variables	0	🟥

Planned Actions

Planned Actions	Importance
Complete the Qlikview dashboard	High
Regression testing of all previous functionality after every run	High
Test system end to end	High
Analyse the incoming data in R and link to the application	Medium
Update progress charts and RAID logs	Low

RAID Log

Risks

Risk Ref	Risk Category	Risk Description	Raised by	Date Identified	Priority	Impact	Prob	Mitigation Category	Mitigation	Owner
R01	technology	No data backup available	BG	01-Mar-14	H	H	L	prevention	Backup all files to Google Drive	BG
R02	Learning	Not enough covered in data mining class to adapt R functionality	BG	01-Mar-14	M	M	H	acceptance	Try to learn enough techniques in own time	BG
R03	Learning	Expiry of free trial software	BG	01-Mar-14	M	M	L	contingency	Work in college instead of home	BG

Assumptions

Ref #	Assumption	Importance	Certainty	Influence	Test	Test Date
A02	No major injuries/illness will be incurred by developer	4 - critical	1 - unknown	H	N/A	-

Closed Issues

Issue Ref	Issue Description	Raised by	Date Raised	Impact	Priority	Action Plan	Status	Owner	Target Resolution Date	Actual Resolution Date
101	Error thrown when running the python script	BG	27-Feb-14	H	H	Identify the issue using a try-except block in python script	closed	BG	07-Mar-14	04-Mar-14

102	Not all players were being added to the data file	BG	30/2/14	H	H	Output all of the players who were not being returned to a seperate error file and check for any consistencies	closed	BG	10-Mar-14	07-Mar-14
103	Error thrown when loading txt file into mysql	BG	30/2/14	H	H	Search internet for cause of issue	closed	BG	10-Mar-14	07-Mar-14
104	Errors thrown when scraping the website for the fixture list	BG	29/3/14	H	H	Check for solutions on stack overflow	Closed	BG	30/3/14	30/3/14

Dependencies

Dependency Ref	Project	Dependency Description	Raised by	Date Raised	Impact	Priority	Period Affected	Action Plan	Owner	Target Resolution Date	Actual Resolution Date
D001	Company Facilities	IT facilities available for running model	BG	26-Feb-14	M	H	Feb - May	Download appropriate free software	BG	Mar-14	Mar-14

Conclusions and Suggestions

At this moment in time the project is well on schedule to be fully completed by the set deadline. Any risks or issues outlined in this document should be monitored on an ongoing basis throughout the project to ensure they do not adversely affect the completion date. The RAID log should be kept fully up to date to enable the effective management of risks and issues and these should then be assigned to the appropriate stakeholder and addressed as appropriate.

No additional resources are required for the project at this time.

Progress Management Report 3

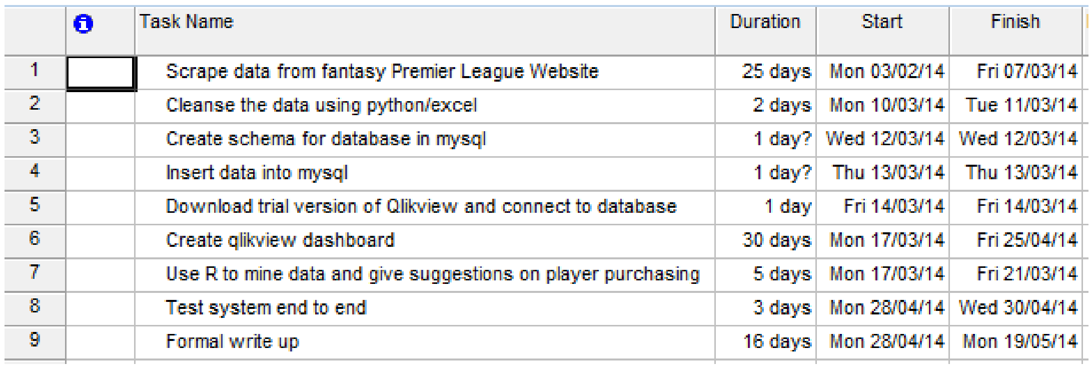
Purpose of Document

The purpose of this document is to give an update to all relevant stakeholders on the project status. This will give the stakeholders a chance to voice their concerns/issues with the progress of the project or any other issues or concerns they may have. The outcome of the report will dictate future activities as, based on discussions, some functionality may be added, removed or adapted.

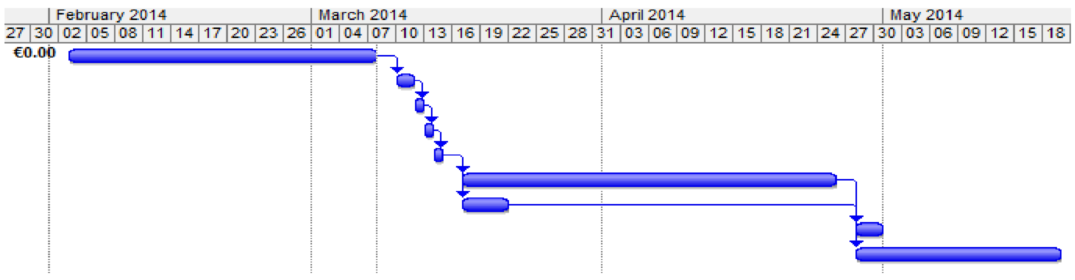
The document should also provide an idea as to the resource requirements for the remainder of the project in order to wait on programme.

Project Timeline

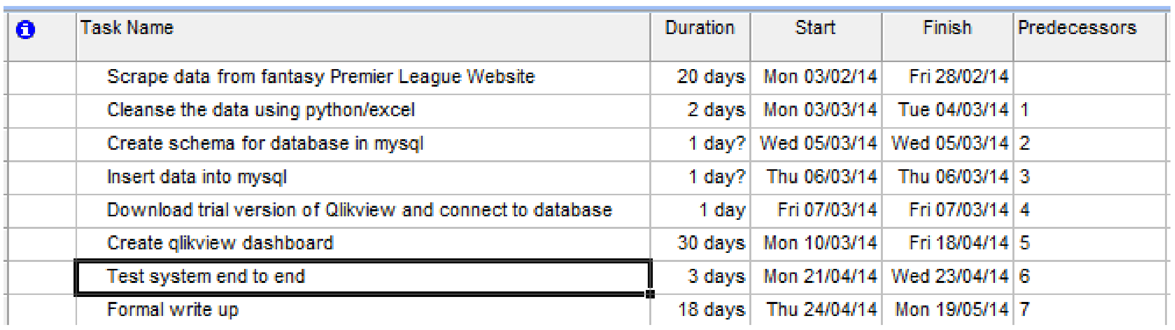
The diagrams below show the original project plan from the Project Proposal document (Gibbons, 2014).



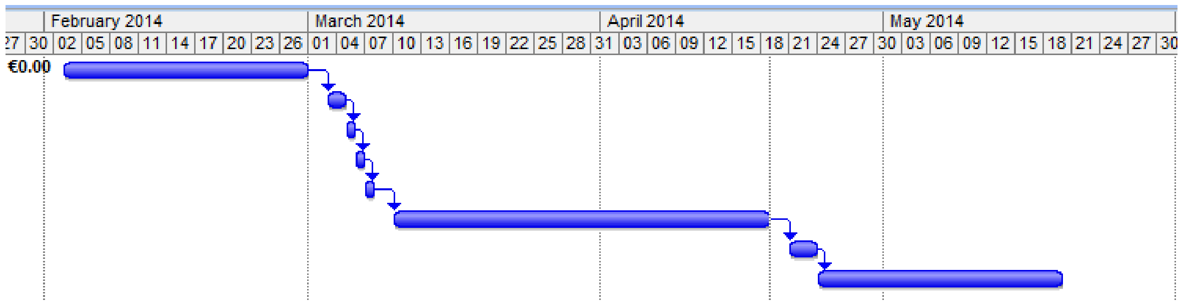
	ℹ	Task Name	Duration	Start	Finish
1		Scrape data from fantasy Premier League Website	25 days	Mon 03/02/14	Fri 07/03/14
2		Cleanse the data using python/excel	2 days	Mon 10/03/14	Tue 11/03/14
3		Create schema for database in mysql	1 day?	Wed 12/03/14	Wed 12/03/14
4		Insert data into mysql	1 day?	Thu 13/03/14	Thu 13/03/14
5		Download trial version of Qlikview and connect to database	1 day	Fri 14/03/14	Fri 14/03/14
6		Create qlikview dashboard	30 days	Mon 17/03/14	Fri 25/04/14
7		Use R to mine data and give suggestions on player purchasing	5 days	Mon 17/03/14	Fri 21/03/14
8		Test system end to end	3 days	Mon 28/04/14	Wed 30/04/14
9		Formal write up	16 days	Mon 28/04/14	Mon 19/05/14



The below diagrams show the updated project plan at this stage in the project:



	Task Name	Duration	Start	Finish	Predecessors
	Scrape data from fantasy Premier League Website	20 days	Mon 03/02/14	Fri 28/02/14	
	Cleanse the data using python/excel	2 days	Mon 03/03/14	Tue 04/03/14	1
	Create schema for database in mysql	1 day?	Wed 05/03/14	Wed 05/03/14	2
	Insert data into mysql	1 day?	Thu 06/03/14	Thu 06/03/14	3
	Download trial version of Qlikview and connect to database	1 day	Fri 07/03/14	Fri 07/03/14	4
	Create qlikview dashboard	30 days	Mon 10/03/14	Fri 18/04/14	5
	Test system end to end	3 days	Mon 21/04/14	Wed 23/04/14	6
	Formal write up	18 days	Thu 24/04/14	Mon 19/05/14	7



Note that the since the last report, it was decided to remove the application of mining the data in R from the scope of this release. The reason for this is explained below.

Project Requirements Status

The below chart indicates the status of each of the current requirements. **The status will be green if it is on schedule, amber if there is a risk of it going off schedule and red if it is currently off schedule.** This information was taken from the Requirements Specification (Gibbons, 2014)

Requirement Number	Requirement	% Complete	Status
1	**Web Scraping Programme**		
1.1	Python programme must scrape data from Fantasy Premier League website for all players in the game	100	🟩
1.2	Programme must be able to interpret .json file types	100	🟩
1.3	Scraping programme must be run after every gameweek has completed and all bonus points updated	100	🟩
1.4	Ouput file from scraping programme is to be a single csv file with all player info	100	🟩
1.5	Programme must be ran from the command line prompt	100	🟩
2	**Mysql Database**		

2.1	The csv file must be loaded into a mysql database	100	🟩
2.2	The mysql tablesand data load should be ran together in one script.	100	🟩
3	**Qlikview Dashboard**		
3.1	The Qlikview application should be downloaded and connected to the mysql database	100	🟩
3.2	The data from the mysql database must be loaded into the qlikview environment using relevant code	100	🟩
3.3	Graphs and tables of player statistics and trendlines must be created within the qlikview application. The code for each of these graphs/tables to be generated independently	50	🟩
3.4	All of the charts graphs must be exportable to excel	50	🟩
3.5	User cannot make amendments to dashboard. Can only view and select the data	0	🟩
3.6	Full end to end system test and UAT on the dashboard scraping -> databse -> dashboard	20	🟨
4.1	A model should be created in R which will interpret input variables for each player and output suggestions based on which players are good/bad picks at that time based on the underlying variables	0	🟥

Due to the tight time constraints it has been decided that action 4.1 is not in scope for this version of the project. It was decided that a fully tested and fully functional dashboard was a higher priority than attempting to rush through the data mining aspect and take time away from system and UAT test.

Action 4.1 has been delayed until the next release of the project

Planned Actions

Planned Actions	**Importance**
Regression testing of all previous functionality after every run	High
Test system end to end	High
Update progress charts and RAID logs	Low
Complete formal write up of the project	High

RAID Log

Risks

Risk Ref	Risk Category	Risk Description	Raised by	Date Identified	Priority	Impact	Prob	Mitigation Category	Mitigation	Owner
R01	technology	No data backup available	BG	01-Mar-14	H	H	L	prevention	Backup all files to Google Drive	BG
R02	Learning	Not enough covered in data mining class to adapt R functionality	BG	01-Mar-14	M	M	H	acceptance	Due to time constraints this functionality has been pushed out to release 2	BG
R03	Learning	Expiry of free trial software	BG	01-Mar-14	M	M	L	contingency	Work in college/work instead of home. The office at work has the fully licensed software so this should not be an issue	BG

Assumptions

Ref #	Assumption	Importance	Certainty	Influence	Test	Test Date
A01	No major injuries/illness will be incurred by developer	4 - critical	1 - unknown	H	N/A	-
A02	All code etc. Has been backed up to numerous locations to mitigate against file corruption/damage to laptop etc	4 - critical	1 - unknown	H	Save all documents to external hard drive and the cloud	

Closed Issues

Issue Ref	Issue Description	Raised by	Date Raised	Impact	Priority	Action Plan	Status	Owner	Target Resolution Date	Actual Resolution Date
101	Error thrown when running the python script	BG	27-Feb-14	H	H	Identify the issue using a try-except block in python script	closed	BG	07-Mar-14	04-Mar-14
102	Not all players were being added to the data file	BG	30/2/14	H	H	Output all of the players who were not being returned to a seperate error file and check for any consistencies	closed	BG	10-Mar-14	07-Mar-14
103	Error thrown when loading txt file into mysql	BG	30/2/14	H	H	Search internet for cause of issue	closed	BG	10-Mar-14	07-Mar-14
104	Errors thrown when scraping the website for the fixture list	BG	29/3/14	H	H	Check for solutions on stack overflow	Closed	BG	30/3/14	30/3/14
105	Duplication of records when importing data to qlikview	BG	10/4/14	H	H	Check for solutions on qlikview community. Associative join in qlikview not the same as inner joins in SQL (Qlikview, 2014)	Closed	BG	11/4/14	11/4/14

Dependencies

Dependency Ref	Project	Dependency Description	Raised by	Date Raised	Impact	Priority	Period Affected	Action Plan	Owner	Target Resolution Date	Actual Resolution Date
D001	Company Facilities	IT facilities available for running model	BG	26-Feb-14	M	H	Feb - May	Download appropriate free software	BG	Mar-14	Mar-14

Conclusions and Suggestions

At this moment in time the project is well on schedule to be fully completed by the set deadline. Any risks or issues outlined in this document should be monitored on an ongoing basis throughout the project to ensure they do not adversely affect the completion date. The RAID log should be kept fully up to date to enable the effective management of risks and issues and these should then be assigned to the appropriate stakeholder and addressed as appropriate.

No additional resources are required for the project at this time.

Appendix E – Python Web Scraping Code

```
import json
import urllib
import re
import io
from bs4 import BeautifulSoup
import os

########################################################################
#           Extract all of the player based information from the website                              #
########################################################################

i = 1
#Open the player file and make it writable
myfile = open("player_history.txt", "w")
myfile.close()

#Create a file to contain all the numbers for which there was errors.
errfile = open("errfile.txt", "w")
errfile.close()

#Website from which to scrape
while i < 700:
        htmltext = urllib.urlopen("http://fantasy.premierleague.com/web/api/elements/" + str(i) +
"/")

        #Use a try-except block to ignore htmls that do not relate to players
        try:
                #Use the json command to read in the json file
                data = json.load(htmltext)
                #Extract the score history from the json file
                scoredata = data["fixture_history"]["all"]
                #Extract the player names
                playerdata = data["first_name"] + " " + data["second_name"]
                #Extract player team
                teamname = data["team_name"]
                #Extract player position
                position = data["type_name"]
                #Extract the players price
                price = data["event_cost"]
                #Percentage selected
                selected = data["selected_by"]
                #Open the file using the io.open with encoding='utf8' to counteract irregualr
characters
                myfile = io.open("player_history.txt", "a", encoding='utf8')

                #Append the data to the file
                for datapoint in scoredata:
                        mystring = str(datapoint)
                        #Clean the data strings
                        mystring1 = mystring.replace("[", "")
                        mystring2 = mystring1.replace("u'", "")
                        mystring3 = mystring2.replace("]", "")
                        mystring4 = mystring3.replace("'", "")
                        #Write the data to the file
                        myfile.write(mystring4 + "," + playerdata + "," + teamname + "," + position +
"," + selected + "," + str(price) + ',' + str(i)  + "\n")
        except:
                #Write all of the numbers for which there was errors to a file
                errfile = open("errfile.txt", "a")
                errfile.write(str(i) + "\n")
                pass

        print i
        i += 1

print "Player data scraped"
#####################################################################
#                       Extract all of the fixture and result information from the website                    #
#####################################################################

base = "http://fantasy.premierleague.com/fixtures/"

#Create a loop to run through the 38 gameweeks
week = 1
fix_file = open('fixtures.txt', 'w')
while week < 39:
        myurl = base + str(week)
        html = urllib.urlopen(myurl).read()
        soup = BeautifulSoup(html)
        fixture_table = soup.find("table", {"class":"ismFixtureTable"})
        #scrape the website for the games played and points
        hometeam = soup.findAll("td", {"class":"ismHomeTeam"})
        awayteam = soup.findAll("td", {"class":"ismAwayTeam"})
        score = soup.findAll("td", {"class":"ismScore"})
                        i = 0
        #Keep looping until the code fails
        while True:
                try:
                        fix_file.write(str(week) + ',' + hometeam[i].text + ',' + score[i].text + ',' +
awayteam[i].text.strip() + ',' + '\n')
                        i += 1
                except:
                        break
        print week
        week += 1

fix_file.close()

print "Fixture data scraped"

###############################################
#       Extract the league table from the website               #
###############################################

base = "http://fantasy.premierleague.com/transfers/"

html = urllib.urlopen(base).read()
soup = BeautifulSoup(html)
#scrape the website for the games played and points
games_played = soup.findAll("td", {"class":"col-pld"})
points = soup.findAll("td", {"class":"col-pts"})
#create an empty list for team names
team = []

#Find all of the team names and append them to the list
for text in soup.find_all('table', {"class":'leagueTable'}):
        for links in text.find_all('a'):
                team.append(links.text.strip())

league_table = open("league_table.txt", "w")
league_table.close()

league_table = open("league_table.txt", "a")

#print out the league table
i=0
while i < 20:
        league_table.write(str(i+1) + "," + team[i] + "," + games_played[i].text + "," + points[i].text +
"\n")
```

```
        i +=1

league_table.close()

print "League table scraped"
print "All data scraped"
```

Appendix F – MySQL Code

```
DROP DATABASE PROJECT;
create database PROJECT;
use PROJECT;

/*CREATE THE TABLE WITH THE INFO FROM THE JSON FILE FROM PYTHON*/
CREATE TABLE PLAYER_STATS(

GAME_DTE                VARCHAR(30),
GAMEWEEK                INTEGER,
OPPOSITION              VARCHAR(30),
MINS_PLYD               INTEGER,
GOALS_SCORED            INTEGER,
ASSISTS                 INTEGER,
CLEAN_SHEET             INTEGER,
GOALS_CONCEDED          INTEGER,
OWN_GOALS               INTEGER,
PENALTIES_SAVED         INTEGER,
PENALTIES_MISSED        INTEGER,
YELLOW_CARDS            INTEGER,
RED_CARDS               INTEGER,
SAVES                   INTEGER,
BONUS                   INTEGER,
EA_PPI                  INTEGER,
BONUS_POINTS_SYS        INTEGER,
NET_TRANSFERS           INTEGER,
PLAYER_VALUE            INTEGER,
POINTS                  INTEGER,
PLAYER_NAME             VARCHAR(50),
TEAM_NAME               VARCHAR(30),
POSITION                VARCHAR(30),
SELECTED_BY             VARCHAR(5),
PRICE                   INTEGER,
PLAYER_ID               INTEGER
)
;

LOAD DATA LOCAL INFILE "C:/Users/Gibbo/Documents/Data
Analytics/Project/Python/player_history.txt"
INTO TABLE PLAYER_STATS
COLUMNS TERMINATED BY ','
;
```

```
/*DELETE THE ENTRY RELATING TO GARETH BALE*/
delete from PLAYER_STATS where player_name = 'Gareth Bale'
;

/*MODIFY THE OPPOSITION FIELD FROM THE PLAYER_STATS TABLE TO SHOW RESULTS IN CORRECT
FORMAT*/

CREATE TABLE PLAYER_HISTORY
AS
(
SELECT
GAME_DTE,
GAMEWEEK,
SUBSTRING(TRIM(OPPOSITION), 1, 3) AS OPPOSITION,
SUBSTRING(TRIM(OPPOSITION), 5, 1) AS HOME_AWAY,
SUBSTRING(TRIM(OPPOSITION), 8, 3) AS SCORE,

CASE  WHEN  SUBSTRING(TRIM(OPPOSITION),  8,  1)  >  SUBSTRING(TRIM(OPPOSITION),  10,  1)  THEN
'WIN'
          WHEN SUBSTRING(TRIM(OPPOSITION), 8, 1) < SUBSTRING(TRIM(OPPOSITION), 10, 1) THEN
'LOSE'
          ELSE 'DRAW'
END AS MATCH_OUTCOME,

MINS_PLYD                    ,
GOALS_SCORED                 ,
ASSISTS                      ,
CLEAN_SHEET                  ,
GOALS_CONCEDED               ,
OWN_GOALS                    ,
PENALTIES_SAVED              ,
PENALTIES_MISSED             ,
YELLOW_CARDS                 ,
RED_CARDS                    ,
SAVES                        ,
BONUS                        ,
EA_PPI                       ,
BONUS_POINTS_SYS             ,
NET_TRANSFERS                ,
PLAYER_VALUE                 ,
POINTS                       ,
PLAYER_NAME                  ,
TEAM_NAME                    ,
POSITION                              ,
CAST(SELECTED_BY AS DECIMAL(3,1)) AS SELECTED_BY,
PRICE                                 ,
PLAYER_ID

FROM PLAYER_STATS
)
;

/*CREATE THE LEAGUE TABLE TABLE*/
CREATE TABLE LEAGUE_TABLE
(
POSITION                     INTEGER,
TEAM_NAME                    VARCHAR(30),
GAMES_PLAYED                 INTEGER,
POINTS                       INTEGER
)
;

LOAD DATA LOCAL INFILE "C:/Users/Gibbo/Documents/Data
Analytics/Project/Python/league_table.txt"
INTO TABLE LEAGUE_TABLE
COLUMNS TERMINATED BY ','
;

/*CREATE THE FIXTURES/RESULTS TABLE SCRAPED FROM PYTHON*/
CREATE TABLE FIXTURES
(
GAMEWEEK                     INTEGER,
HOME_TEAM                    VARCHAR(30),
SCORE                        VARCHAR(10),
AWAY_TEAM                    VARCHAR(30)
)
;

LOAD DATA LOCAL INFILE "C:/Users/Gibbo/Documents/Data Analytics/Project/Python/fixtures.txt"
INTO TABLE FIXTURES
COLUMNS TERMINATED BY ','
;

/*MODIFY THE ABOVE TABLE TO SHOW GAMES PLAYED AND REMAINING GAMES*/
CREATE TABLE FIXTURE_STATUS
AS
(
SELECT GAMEWEEK,
                    HOME_TEAM,
                    SCORE,
                    AWAY_TEAM,
                    CASE WHEN SCORE LIKE ('%0%')
                              OR SCORE LIKE ('%1%')
                              OR SCORE LIKE ('%2%')
                              OR SCORE LIKE ('%3%')
                              OR SCORE LIKE ('%4%')
                              OR SCORE LIKE ('%5%')
                              OR SCORE LIKE ('%6%')
                              OR SCORE LIKE ('%7%')
                              OR SCORE LIKE ('%8%')
                              OR SCORE LIKE ('%9%') THEN 'RESULT' ELSE 'TBP' END AS STATUS

FROM FIXTURES
)
;


CREATE TABLE TEAM(
TEAM_NAME                    VARCHAR(30),
TEAM_ABBRV                   CHAR(3)
)
;

INSERT INTO TEAM VALUES('Arsenal', 'ARS');
INSERT INTO TEAM VALUES('Crystal Palace', 'CRY');
INSERT INTO TEAM VALUES('Aston Villa', 'AVL');
INSERT INTO TEAM VALUES('Stoke City', 'STK');
INSERT INTO TEAM VALUES('Fulham', 'FUL');
INSERT INTO TEAM VALUES('Cardiff City', 'CAR');
INSERT INTO TEAM VALUES('Chelsea', 'CHE');
INSERT INTO TEAM VALUES('Man Utd', 'MUN');
INSERT INTO TEAM VALUES('Liverpool', 'LIV');
INSERT INTO TEAM VALUES('Everton', 'EVE');
INSERT INTO TEAM VALUES('West Brom', 'WBA');
INSERT INTO TEAM VALUES('Hull City', 'HUL');
INSERT INTO TEAM VALUES('West Ham', 'WHU');
INSERT INTO TEAM VALUES('Sunderland', 'SUN');
INSERT INTO TEAM VALUES('Man City', 'MCI');
INSERT INTO TEAM VALUES('Newcastle', 'NEW');
INSERT INTO TEAM VALUES('Norwich', 'NOR');
INSERT INTO TEAM VALUES('Southampton', 'SOU');
INSERT INTO TEAM VALUES('Swansea', 'SWA');
INSERT INTO TEAM VALUES('Tottenham', 'TOT');


/*TEAM LEVEL VIEW OF THE REMAINING FIXTURES*/
CREATE TABLE FIXTURES_REMAINING
AS
(
SELECT FIX.GAMEWEEK,
                TM.TEAM_NAME,
                CASE  WHEN  TM.TEAM_NAME  =  FIX.HOME_TEAM  THEN  AWAY_TEAM  ELSE
HOME_TEAM END AS OPPOSITION,
                CASE  WHEN  TM.TEAM_NAME  =  FIX.HOME_TEAM  THEN  'H'  ELSE  'A'  END  AS
HOME_OR_AWAY



FROM FIXTURES                                         FIX

LEFT JOIN TEAM                                        TM
ON TM.TEAM_NAME = FIX.HOME_TEAM
OR TM.TEAM_NAME = FIX.AWAY_TEAM

WHERE SCORE LIKE '%V%'
)
;

CREATE TABLE PLAYER_POINTS
AS
(
SELECT PLAYER_NAME
                ,SUM(POINTS) AS TOT_POINTS

FROM PLAYER_STATS

GROUP BY 1
)
;
```

Appendix G – Qlikview Data Load Code

```
SET ThousandSep=',';
SET DecimalSep='.';
SET MoneyThousandSep=',';
SET MoneyDecimalSep='.';
SET MoneyFormat='€#,##0.00;-€#,##0.00';
SET TimeFormat='hh:mm:ss';
SET DateFormat='DD/MM/YYYY';
SET TimestampFormat='DD/MM/YYYY hh:mm:ss[.fff]';
SET MonthNames='Jan;Feb;Mar;Apr;May;Jun;Jul;Aug;Sep;Oct;Nov;Dec';
SET DayNames='Mon;Tue;Wed;Thu;Fri;Sat;Sun';

ODBC CONNECT TO Test;

TEAM:
LOAD
TEAM_NAME,
TEAM_ABBRV
;

SQL SELECT *
FROM TEAM
;

LEAGUE:
LOAD
POSITION AS TEAM_POSITION,
//Modify the team names to match the values in the player history table
IF(MATCH(TEAM_NAME, 'Cardiff', 'Stoke', 'Hull'),
       TEAM_NAME & ' City',
       IF(TEAM_NAME = 'Spurs', 'Tottenham',
              TEAM_NAME)) AS TEAM_NAME,
GAMES_PLAYED,
POINTS AS TEAM_POINTS
;
SQL SELECT *
FROM LEAGUE_TABLE
;



PLAYER:
LOAD
GAME_DTE,
GAMEWEEK,
LOOKUP('TEAM_NAME', 'TEAM_ABBRV', OPPOSITION, 'TEAM') AS OPPOSITION_NAME,
LOOKUP('TEAM_POSITION', 'TEAM_NAME', LOOKUP('TEAM_NAME', 'TEAM_ABBRV', OPPOSITION,
'TEAM'), 'LEAGUE') AS OPPOSITION_POSITION,
HOME_AWAY,
SCORE as RESULT,
MATCH_OUTCOME,
MINS_PLYD,
GOALS_SCORED,
ASSISTS,
CLEAN_SHEET,
GOALS_CONCEDED,
OWN_GOALS,
PENALTIES_SAVED,
PENALTIES_MISSED,
YELLOW_CARDS,
RED_CARDS,
SAVES,
BONUS,
EA_PPI,
BONUS_POINTS_SYS,
NET_TRANSFERS,
PLAYER_VALUE/ 10 as PLAYER_VALUE,        //Value appears as a multiple of 10 in the
file
POINTS,
PLAYER_NAME,
TEAM_NAME,
POSITION,
PRICE/10 as PRICE,
SELECTED_BY,
PLAYER_ID
;
SQL SELECT *
FROM PLAYER_HISTORY
;


FIXTURES:
LOAD
GAMEWEEK AS GAMEWEEK_NO,
HOME_TEAM & ' v ' & AWAY_TEAM AS FIXTURE,
HOME_TEAM,
SCORE,
AWAY_TEAM,
STATUS
;

SQL SELECT *
FROM FIXTURE_STATUS
;

FIXTURES_REMAINING:
LOAD
GAMEWEEK AS GAMEWEEK_NO,
TEAM_NAME,
OPPOSITION,
HOME_OR_AWAY
;

SQL SELECT *
FROM FIXTURES_REMAINING
;




POINTS:
LOAD
PLAYER_NAME,
TOT_POINTS
;

SQL SELECT *
FROM PLAYER_POINTS
;


Table:
Load * inline
[
No. Players
5,
10,
15,
20
]
;
```

Appendix H – Dashboard Regression Test Plan

Test Case ID	Area	Description	Expected Results	Actual Results	Pass/fail	Tester	Date
D1	Dashboard	League table is correct as at the date the dashboard was loaded	League Table calculated in the dashboard should match the actual league table	Calculated table matches real table from the website	Pass	BG	28/03/2014
D2	Dashboard	Player points data matches that of the website	Points data in the dashboard should match the points given in the website	Points match	Pass	BG	28/03/2014
D3	Dashboard	All players are available for selection in the dashboard	Every player that appears in the website should be available in the dashboard	All players are available	Pass	BG	28/03/2014
D4	Dashboard	Information for each player should be correct eg (price, goals scored, assists, yellow cards etc)	Information should be correct	Information is correct as per the website	Pass	BG	28/03/2014
D5	Dashboard	When a team is selected, only players from that team should be available in the player selection box	1. Select a team 2. Look at the player selecton box 3. Only players from that team should be available for further selection	When a team is selected only players from that team can be selected	Pass	BG	28/03/2014
D6	Dashboard	When a position is selected, only players in that position should be available in the player selection box	1. Select a position 2. Look at the player selecton box 3. Only players from that position should be available for further selection	When a position is selected only players from that team can be selected	Pass	BG	28/03/2014

D7	Dashboard	Tab names should be consistent across sheets	1. Select a sheet by clicking on a tab. 2. The names of the tabs should be the same across all sheets	Names of tabs are consistent	Pass	BG	28/03/2014
D8	Dashboard	When a tab is selected, the tab should change to a bright blue colour	1. Select a tab 2. the tab should change to a bright blue colour	The tab changes to a bright blue colour	Pass	BG	28/03/2014
D9	Dashboard	The drop down box for the moving average graph should allow you to change the number of gameweeks the moving average relates to	1. Select the Moving Average tab in the Player Data sheet 2. Select a player 3. Select the number of steps from the drop down 4. The result should be that number of steps moving average	Ok for 1 - 6 steps	Pass	BG	28/03/2014
D10	Dashboard	When a value is selected from the dropdown only players at or below that value should be available	1. Select a value from the dropdown 2. Check the player selection box 3. Only players within that value should be available for selection	Only players at or below the value are returned	Pass	BG	28/03/2014
D11	Dashboard	Graphs and tables can be exported to excel	1. Click the XL icon on any table or graph 2. The data should be exported to an excel file	Data is exported to an excel file	Pass	BG	28/03/2014

Appendix I – Link to Google Drive Folder with Data

https://drive.google.com/folderview?id=0Bx62KS94j6uhM18tUjNfRDdsQVE&usp=sharing